Pablo Rauzy

✉ pablo.rauzy@inria.fr – r@uzy.me
http://pablo.rauzy.name/
26 ans, français

Candidature – *Qualification aux fonctions de Maître de Conférence*

Composition du dossier

- Déclaration de candidature [2 pages]
- CV [2 pages]
- Présentation de mes travaux antérieurs d'enseignement et de recherche [4 pages]
- Liste de publications (celles avec ⋆ sont jointes au dossier) [2 pages]

- Diplôme thèse :
  - Attestation du diplôme de thèse [1 page]
    (Télécom ParisTech ne délivrera pas le diplôme lui-même avant mars 2016)
  - Procès verbal de soutenance de thèse [3 pages]
  - Rapport de pré-soutenance de Pierre-Alain Fouque [3 pages]
  - Rapport de pré-soutenance de Marie-Laure Potet [3 pages]

- Thèse :
  - Avis de Sylvain Guilley (directeur de thèse) [2 pages]
  - Contrat doctoral + mission d'enseignement [7 pages]

- Mission doctorale d'enseignement :
  - Avis de Nizar Ouarti (responsable du cours de C) [1 page]
  - Avis de Hacène Ouzia (responsable du cours de développement web) [1 page]
  - Avis de François Pécheux (responsable enseignements informatique à Polytech'UPMC) [1 page]
  - Attestation de service enseignement [1 page]
  - Attestation d'encadrement du stage de M2 de Martin Moreau [1 page]

- Post-doc :
  - Avis de Daniel Le Métayer (encadrant de post-doc) [1 page]
  - Contrat de travail post-doc [4 pages]

- Publications jointes :
  - Formally Proved Security of Assembly Code Against Power Analysis [26 pages]
  - Countermeasures Against High-Order Fault-Injection Attacks on CRT-RSA [23 pages]
  - Using Modular Extension to Provably Protect ECC Against Fault Attacks [39 pages]

Il est conseillé de joindre ce document au dossier transmis
aux rapporteurs désignés par le Conseil National des Universités

DÉCLARATION DE CANDIDATURE À LA QUALIFICATION
AUX FONCTIONS DE MAÎTRE DE CONFÉRENCES,
POUR LA SECTION 27-Informatique
(Campagne 2016)

Je soussigné(e) M.
Nom de famille : RUZY
Nom d'usage : RAUZY
Prénom : PABLO
Date et lieu de naissance : 30/07/1989 - MARSEILLE 8E
Nationalité : Française

Adresse postale et électronique à laquelle seront acheminées toutes les correspondances
adresse Code postal : 69100 Ville :VILLEURBANNEP ays : RANCE Téléphone : tel Télécopie : Adresse électronique : mail

Date de création de la candidature
11/09/2015 à 00:09

Date de dernière modification de la candidature
23/09/2015 à 11:09

Titres universitaires français :
Doctorat

Diplôme au titre duquel la qualification est demandée : Doctorat
Titre : Méthodes logicielles formelles pour la sécurité des implémentations cryptographiques
Date de soutenance : 13/07/2015
Lieu de la soutenance : TELECOM PARISTECH
Mention : très honorable
Directeur : SYLVAIN GUILLEY
Composition du jury : PIERRE-ALAIN FOUQUE
MARIE-LAURE POTET
FRANCOIS DUPRESSOIR
KARINE HEYDEMANN
MEDHI TIBOUCHI
DAVID NACCACHE

Liste des établs et labos d'exercice :
Post-doctorat (situation actuelle) :
* Recherche au CITI (EA 3720 Inria et INSA Lyon), dans l'équipe Privatics.

Pendant la thèse :
* Recherche au LTCI (UMR 5141 Télécom ParisTech et CNRS), dans l'équipe SEN.
* Enseignement à Polytech Paris-UPMC

Activités en matière d'enseignement :
* 32h + 32h de TP, niveau L3, "programmation en C".
* 32h + 40h de TP, niveau L3, "développement web (HTML, CSS, PHP, MySQL)".
* 54h UE complète (cours magistraux, TP, examens papier et machine, projet), niveau M1, "programmation orienté objet et langage C++".
* Co-encadrement d'un stage de M2 recherche (6 mois).

Thème de recherche et mots clés :
Codes 32, 62, 63, 65, 73 : cryptographie, approches formelles, langages, systèmes embarqués, sécurité, preuve.

En thèse : approches formelles de l'étude et la conception de contre-mesures aux attaques physiques (par canaux auxiliaires ou par injections de fautes) sur les systèmes cryptographiques.

En post-docorat : approches formelles de la protection de la vie privée.

Activités en matière d'administration et autres responsabilités collectives :
Avant la thèse : élu représentant des étudiants informatique de l'ENS pendant 3 ans, et élu représentant des étudiants scientifiques de l'ENS au conseil d'administration de l'ENS pendant 1 an.

Pendant la thèse : gestion d'une UE à Polytech'UPMC.

déclare faire acte de candidature à la qualification.
Fait à Lyon **le** 10/12/2015

Signature

Pablo Rauzy

✉ *pablo.rauzyXXXinria.fr*
http://pablo.rauzy.name/

Curriculum vitæ – *Qualification aux fonctions de Maître de Conférence*

Naissance : le 30 juillet 1989, à Marseille, nationalité française.
Adresse laboratoire : Laboratoire CITI (INSA Lyon / Inria),
56 bd Niels Bohr (CEI2),
69100 Villeurbanne.
Téléphone laboratoire : +33(0)4 XX XX XX XX
Adresse personnelle : XX XX XXXXXXX,
69100 Villeurbanne
Téléphone personnel : +33(0)6 XX XX XX XX

Parcours académique

2007 **Baccalauréat** série S option Sciences de l'ingénieur et spécialité Mathématiques, mention assez bien, lycée Marseilleveyre (Marseille).

2007 - 2009 **DEUG** Mathématiques et Informatique, mention félicitation du jury, Université de la Méditerranée (Marseille).

2009 - 2010 **Licence** Informatique, mention bien, École normale supérieure (Paris).
- *Stage L3* de 3 mois avec Christophe Rippert, Karin Altisen, et Kévin Marquet au Vérimag dans l'équipe Synchrone, mémoire intitulé "A formal approach to the development of system services in embedded systems".

2010 - 2012 **Master** Parisien de Recherche en Informatique, École normale supérieure (Paris).
- *Stage M1* de 5 mois avec Clemens Grelck à l'Université d'Amsterdam dans le Computer System Architecture group, mémoire intitulé "Implicit parallelization of code called from an external and already parallelized environment".
- *Stage M2* de 6 mois avec Marc Pouzet à l'ENS dans l'équipe Parkas, mémoire intitulé "Mixing continuous and discrete time in a synchronous language".

2009 - 2012 **Diplôme de l'ENS**, École normale supérieure (Paris). Département informatique.

oct 2012 - sept 2015 **Thèse** dans l'équipe SEN (Sécurité Électronique et Numérique) du département COMELEC (Communication et Électronique) de Télécom ParisTech (Paris).
- Titre : "Méthodes logicielles formelles pour la sécurité des implémentations de systèmes cryptographiques".
- Laboratoire : LTCI (UMR 5141)
- Directeur de thèse : Sylvain Guilley, Professeur, Télécom ParisTech.
- Soutenue le 13 juillet 2015 à l'École normale supérieure (mention très honorable).
- Composition du jury :
  - François Dupressoir, Chercheur, IMDEA Software Institute (examinateur),
  - Pierre-Alain Fouque, Professeur, Univ. Rennes 1 (rapporteur),
  - Karine Heydemann, Maître de Conférence, UPMC (examinatrice),
  - David Naccache, Professeur, Univ. Panthéon-Assas (examinateur - président),
  - Marie-Laure Potet, Professeure, Ensimag (rapporteuse),
  - Mehdi Tibouchi, Chercheur, NTT (examinateur),
  - David Vigilant, Chercheur, Gemalto (invité).
- Financement par une bourse de l'école doctorale ÉDITE (contrat doctoral + mission d'enseignement à Polytech'UPMC).
- *Stage* de 1 mois à l'IMDEA (Madrid) dans l'équipe de Gilles Barthe.
- *Co-encadrement* (à 50%) avec Sylvain Guilley du stage de M2 de Martin Moreau, sur la protection de calculs sur courbes elliptiques contre les attaques par injection de fautes ; ce stage a abouti à un article ainsi qu'à un chapitre d'ouvrage. Martin Moreau est maintenant en thèse à l'IMDEA avec Gilles Barthe.

oct 2015 - sept 2016 (situation actuelle) **Post-doc** dans l'équipe-projet Inria Privatics du laboratoire CITI commun à l'INSA Lyon et au centre Inria Grenoble–Rhône-Alpes (Lyon).
- Projet : "Formal Model for Privacy as Control".
- Encadrant : Daniel Le Métayer.
- Financement par l'équipe-projet Inria.

Domaine de recherche

Je travaille sur la *sécurité de l'information*, notamment sur le respect et la protection de la *vie privée*. D'une part, cela m'amène à m'intéresser à tous les niveaux de la sécurité : à la sécurité physique des implémentations de systèmes embarqués, à l'analyse de vulnérabilités des systèmes d'information et des réseaux, à la cryptologie, à la rétro-ingénierie de logiciels malveillants, et aux tests d'intrusion. D'une autre part, cela m'amène aussi à m'intéresser aux systèmes qui manipulent des données personnelles, en particulier aux systèmes distribués tels que les *réseaux intelligents* (smart grid, smart city, etc.), et de manière plus générale à l'*Internet des objets*.

Mon projet de recherche vise la protection de la vie privée par la mise en œuvre d'une *approche offensive* de la sécurité, c'est à dire de garantir cette dernière en cherchant systématiquement les vulnérabilités des systèmes. Pour cela je m'appuie sur une *approche formelle* de la preuve et de l'automatisation des méthodes de protection.

Production scientifique

Revues internationales à comité de lecture	2	Journal of Cryptographic Engineering (×2)
Chapitres d'ouvrage	1	publié chez CRC Press
Conférences internationales à comité de lecture	7	WISTP, IMACC, TGC, PPREW, FDTC, HOST, COSADE
Conférences internationales à comité de lecture sans acte	4	COSADE, PROOFS (×2), TRUDEVICE
Articles soumis ou en préparation	2	
Exposés lors de conférences internationales	6	dont 4 à l'étranger
Exposés lors de conférences nationales	1	à l'étranger
Séminaires	4	+ 3 invitations à venir
Présentations de posters lors de conférences internationales	2	dont 1 à l'étranger
Logiciels	3	dont 2 sont utilisés par des chercheurs et dans l'industrie

Responsabilités collectives

Membre des comités d'organisation des conférences COSADE 2013, COSADE 2014, et PROOFS 2015.
Relecteur pour les conférences COSADE 2014, SPACE 2015, et CARDIS 2015.
Relecteur externe pour le *Journal of Cryptographic Engineering* en 2013.

Enseignement

2012 - 2013
2013 - 2014 **Programmation en C** (32h + 32h) [N. Ouarti]. Encadrement des TP de langage C et programmation modulaire, à Polytech'UPMC (∼15 étudiant·e·s en 3ème année).

2012 - 2013
2013 - 2014 **Développement web** (32h + 40h) [H. Ouzia]. Encadrement des TP de HTML, CSS, PHP, et MySQL, à Polytech'UPMC (∼15 étudiant·e·s en 3ème année).

2013 - 2014 **Introduction à GNU/Linux et à la ligne de commande** (6h). Encadrement du "TP Zéro" de familiarisation avec GNU/Linux, à Polytech'UPMC (∼30 étudiant·e·s en 1ère année).

2014 - 2015 **Programmation orientée objet et langage C++** (54h). Responsable du cours entier (CM, TP, projets, examens machine et papier), à Polytech'UPMC (∼30 étudiant·e·s en 4ème année).

Mes activités d'enseignement sont décrites en détails par la suite.

Compétences techniques

Programmation	OCaml, C, Racket, C++, ASM	Scriptage	Bash, Python, Emacs Lisp
Web	HTML, CSS, JS, PHP, SQL	Graphisme	Inkscape, GIMP
Outils	LaTeX, Emacs, Git, SVN, Tor, I2P	Systèmes	UNIX, GNU/Linux

Langues

Français	langue maternelle	Anglais	lu / écrit / parlé

Mandats et affiliations

depuis 2008 Membre de l'April et de la FSF.

2008 - 2009 Élu représentant des étudiant·e·s en informatique au CÉVU de la Faculté de Sciences de Luminy.

2009 - 2012 Élu représentant des étudiant·e·s en informatique au conseil du Dpt informatique de l'ENS.

2011 - 2012 Élu représentant des étudiant·e·s scientifiques au Conseil d'administration de l'ENS.

depuis 2015 Co-fondateur et membre de l'association CAPSH qui porte le projet Dissemin (pour le développement du libre accès aux résultats de la recherche).

Pablo Rauzy

✉ pablo.rauzy@inria.fr – r@uzy.me
http://pablo.rauzy.name/
26 ans, français

Travaux antérieurs – *Qualification aux fonctions de Maître de Conférence*

Enseignement

Avant la thèse (total 18h)

Proposition et co-encadrement de petits projets de recherche des étudiant·e·s de L3 du cours "informatique scientifique par la pratique" de David Naccache à l'ENS (3×2h d'encadrement pour chaque projet) :
Autonomic (2011-2012) : créer une version fonctionnelle du jeu Nomic, qui permet aux joueurs de faire évoluer les règles du jeu, en se basant sur une construction de Quine permettant au programme de se réécrire lui-même.
Btrsync (2011-2012) : créer une version améliorée de l'outil de synchronisation de dossier `rsync`, optimisée pour transférer le moins de données possible (en contrepartie de calculs plus conséquents au niveau de la source et de la destination de la synchronisation). Le projet a abouti à la rédaction d'un article qui a été présenté à la conférence TGC 2012 [ABB+12].
Paper laundry (2014-2015) : créer un outils permettant de retirer des PDF d'article de recherche les traces de pistages laissées par les maisons d'édition, pour cela étudier les possibilités de stéganographie dans des PDF et étudier la possibilité de "normaliser" la structure d'un PDF.

Mission doctorale d'enseignement à Polytech'UPMC (total 196h)

2012 - 2013 **Programmation en C** (32h + 32h).
2013 - 2014 J'ai été en charge deux ans (32h / an) des séances TP en groupes d'une quinzaine d'étudiant·e·s, pour le cours enseigné par M. Ouarti en 3ème année (équivalent L3) de la filière "Électronique et Informatique parcours Informatique Industrielle". Les TPs ont couvert les bases de la programmation en C : structures de données, pointeurs, gestion de la mémoire, compilation modulaire (Makefile).
La matériel pour ce cours (fiches de TP) a été fourni par M. Ouarti.

2012 - 2013 **Développement web** (32h + 40h).
2013 - 2014 J'ai été en charge deux ans (32h puis 40h) des séances de TP en groupe d'une quinzaine d'étudiant·e·s, pour le cours enseigné par M. Ouzia en 3ème année (équivalent L3) de la filière "Agroalimentaire" (il s'agit d'un cours d'ouverture en informatique). Les TPs ont couvert les bases du développement web et de la gestion de base de données en utilisant HTML, CSS, PHP et MySQL.
Le matériel pour ce cours (fiches de TP) a principalement été fourni par M. Ouzia, à l'exception de deux séances de TP pour lesquelles j'ai rédigé les sujets (consistant à leur faire faire en deux séances un petit clone de Twitter) qui continuent d'être utilisés depuis. Les sujets que j'ai rédigés sont disponibles sur `http://pablo.rauzy.name/teaching.html#epu-tpweb`.

2013 - 2014 **Introduction à GNU/Linux et à la ligne de commande** (6h).
Encadrement de deux groupes (3h par groupe) d'une trentaine d'étudiant·e·s arrivant à Polytech'UPMC pour leur "TP Zéro" de familiarisation avec l'environnement GNU/Linux des salles informatiques de l'établissement, et introduction à la ligne de commande.

2014 - 2015 **Programmation orientée objet et langage C++** (54h).
En troisième année de thèse j'ai demandé à être responsable d'un cours entier, par envie de découvrir tous les aspects de la gestion d'un cours. François Pécheux, responsable des cours d'informatique à Polytech'UPMC, a accepté de me confier le cours de programmation objet et C++ de 4ème année (équivalent M1) de la filière "Électronique et Informatique parcours Systèmes Embarqués" (une trentaine d'étudiant·e·s).
J'ai choisi de ne pas réutiliser le matériel des années précédentes et de proposer un cours entièrement neuf créé par mes soins. Comme son intitulé l'indique, ce cours couvre la programmation en langage C++ : différences avec le C, programmation orientée objet puis spécifiquement basée sur les classes, surcharge des opérateurs, héritages, templates, exceptions, la STL, puis une bibliothèque multimédia (la SFML). Les TPs ont servi à mettre en pratique les nouvelles notions de chaque séance de cours tout en réutilisant les notions vues auparavant. Pour évaluer les étudiant·e·s, je leur ai fait faire un projet de choix libre en groupe de deux ou trois, avec la contrainte que ce soit un jeu graphique, un examen individuel sur machine (TP plus long et noté), ainsi qu'un examen sur papier.
Tout le matériel de cours est disponible sur `http://pablo.rauzy.name/teaching.html#epu-cpp`.

Co-encadrement (50%) d'un stage de M2 recherche

Pendant le dernier semestre de ma thèse j'ai eu l'opportunité de co-encadrer le stage de recherche de Martin Moreau, un étudiant du M2 "Sécurité Fiabilité et Performance du Numérique" de l'UPMC (Paris 6).
Nous lui avons proposé de travailler sur la protection des algorithmes de cryptographie basés sur les courbes elliptiques (ECC) contre les attaques physiques par injection de fautes, en réutilisant la technique dite d'"extension modulaire" introduite par Shamir pour protéger CRT-RSA (c'est à dire RSA optimisé avec le théorème des reste

chinois) contre ce même type d'attaques.
Le stage s'est extrêmement bien déroulé et a abouti à la rédaction d'un article encore en cours de préparation, et d'un chapitre étendant les résultats obtenus sur l'ECC aux calculs de couplages, qui fera partie du livre *Handbook of Pairing Based Cryptography* (Nadia El Mrabet et Marc Joye éditeurs, CRC Press Taylor and Francis group) [RMG16].
Ce stage a été l'occasion pour Martin de confirmer son envie de poursuivre dans la recherche et il est maintenant en thèse avec Gilles Barthe à l'IMDEA Software Institute, à Madrid.

Recherche

Stage L3 : 3 mois dans l'**équipe Synchrone du laboratoire Verimag** à Grenoble avec Christophe Rippert, Karine Altisen et Kévin Marquet. Mémoire : "*une approche formelle du développement de services systèmes dans les systèmes embarqués temps réel*". Résultat principal : développement d'un ordonnanceur dynamique pour le langage Lustre qui conserve les garanties temps réel de l'ordonnancement statique.

Stage M1 : 5 mois dans le **groupe Computer Systems Architecture à l'université d'Amsterdam** avec Clemens Grelck. Mémoire : "*la parallélisation implicite de code appelé depuis un environnement extérieur et déjà parallélisé*". Résultat principal : redéveloppement de l'interface externe C du langage SaC (Single Assignment C, fait pour la parallélisation implicite de calculs sur des tableaux) pour lui permettre de faire de la parallélisation automatique même quand le code C qui s'interface avec le programme SaC contient déjà des threads, ce qui n'était pas possible avant.

Stage M2 : 6 mois dans l'**équipe Inria Parkas à l'ENS Ulm**, avec Marc Pouzet. Mémoire : "*le mélange de temps continu et discret dans un langage synchrone*". Résultat principal : une analyse statique de Zélus, un langage synchrone hybride (mixant temps discret et continu), qui détecte si une séquence infinie d'étapes discrètes peut arriver entre deux phases continues. La présence d'un nombre fini d'étapes discrètes entre deux phases continues est une condition nécessaire pour s'assurer que le temps avance pendant la simulation d'un système.

Thèse

Méthodes logicielles formelles pour la sécurité des implémentations de systèmes cryptographiques

Les implémentations cryptographiques sont vulnérables aux attaques physiques, et ont donc besoin d'en être protégées. Bien sûr, des protections défectueuses sont inutiles. L'utilisation des méthodes formelles permet de développer des systèmes tout en garantissant leur conformité à des spécifications données. Le premier objectif de ma thèse, et son aspect novateur, est de montrer que les méthodes formelles peuvent être utilisées pour prouver non seulement les principes des contre-mesures dans le cadre d'un modèle, mais aussi leurs implémentations, étant donné que c'est là que les vulnérabilités physiques sont exploitées. Mon second objectif est la preuve et l'automatisation des techniques de protection elles-mêmes, car l'écriture manuelle de code est sujette à de nombreuses erreurs, particulièrement lorsqu'il s'agit de code de sécurité.

Les attaques physiques peuvent être classifiées en deux catégories distinctes. Les attaques passives, où l'attaquant peut seulement lire l'information qui fuit par *canaux auxiliaires* (comme la consommation de courant ou les émanations électromagnétiques). Et les attaques actives, où l'attaquant perturbe le système pour faire en sorte de lui faire révéler des secrets via sa sortie standard. Par conséquent, j'ai poursuivi mes objectifs dans ces deux cadres : sur une contre-mesure qui diminue les fuites par canaux auxiliaires, et sur des contre-mesures contre les attaques par *injection de faute*.

Comme il existe déjà des propriétés rigoureuses de sécurité pour les protections contre les fuites par canaux auxiliaires, mes contributions se concentrent sur l'exploitation de méthodes formelles pour la conception et la vérification d'implémentations d'algorithmes protégés. J'ai développé une méthode de protection qui, étant donné une implémentation, en génère une version améliorée qui a un rapport signal à bruit nul sur ses canaux auxiliaires, grâce au fait que la fuite a été rendue constante (en particulier, la fuite ne dépend pas des données sensibles) en utilisant la méthode dite du "double rail" (*dual-rail with precharge logic*). Dans l'intérêt de la démonstration, j'ai aussi entrepris d'écrire un outil (paioli) qui automatise l'application de cette méthode sur un code non sécurisé écrit en langage assembleur. Indépendemment, l'outil permet de prouver que la propriété de fuite constante est toujours vérifiée pour une implémentation donnée, ce qui permet de vérifier systématiquement le résultat de la méthode de protection, mais sert aussi de test de non-régression de sécurité en cas d'optimisation manuel du code obtenu. À ma connaissance, paioli est le premier outil permettant de protéger automatiquement une implémentation contre les fuites par canaux auxiliaires en équilibrant sa fuite de manière prouvable.
→ Les résultats de ce travail ont été présentés lors de la conférence internationale avec comité de lecture mais sans actes PROOFS 2014 (à Busan, en Corée) [RGN14] puis publiés dans une revue internationale avec comité de lecture [RGN15].

À l'inverse, la définition même des objectifs de sécurité n'était pas clairement établie pour les attaques par injection de faute lorsque j'ai commencé ma thèse. Les propriétés de sécurité à prouver n'ayant même pas été formellement énoncées, beaucoup de contre-mesures ont été publiées sans preuve. C'est seulement lors de ma thèse que les "conditions de succès d'attaque" ont été introduites.

En conséquence, la première question a été d'évaluer les contre-mesures existantes par rapport à ces conditions de succès d'attaque. À cette fin, j'ai développé une méthode, basée sur l'évaluation symbolique par réécriture d'arbre

en suivant les règles de l'arithmétique modulaire, qui permet la couverture complète des fautes possibles sur un algorithme (implémentant une contre-mesure) et le calcul de leurs effets. J'ai implémenté cette méthode dans un outil (finja), qui m'a permis (ainsi qu'à d'autres chercheur·e·s) de vérifier et prouver certaines contre-mesures, de retrouver des attaques connues, et aussi d'en découvrir de nouvelles.
→ Les résultats de ce travail ont été présentés lors de la conférence internationale avec comité de lecture mais sans actes PROOFS 2013 (à Santa Barbara, aux États-Unis) [RG13] puis publiés dans une revue internationale avec comité de lecture [RG14a].

La seconde question portait sur la minimalité des contre-mesures. J'ai entre autres étudié en profondeur l'une des contre-mesures de l'état de l'art (développée chez Gemalto par David Vigilant). Le résultat de cette étude formelle utilisant finja a été la simplification drastique de la contre-mesure, aussi bien au niveau de la longueur du code que de la nécessité de nombres aléatoires (qui sont coûteux à générer), et ce sans affecter ses propriétés de sécurité. En effet, avec mes simplifications, la contre-mesure est passée de 9 à 3 vérifications d'invariants et ne nécessite plus qu'un seul nombre aléatoire, contre 5 auparavant. Ce travail a montré la valeur ajoutée de l'approche formelle par rapport à l'ingénierie par essais-erreurs qui a été jusqu'à présent la méthode principale de développement de contre-mesures.
→ Les résultats de ce travail ont été présentés et publiés dans dans la conférence internationale avec comité de lecture PPREW 2014 (à San Diego, aux États-Unis) [RG14b].

Les contre-mesures existantes revendiquent de protéger contre une ou parfois deux fautes. Cependant, des attaques utilisant plus de deux fautes ont vu le jour, aussi bien en pratique qu'en théorie. Cependant, comme finja m'a permis de le découvrir, les contre-mesures se revendiquant du second ordre (résistantes à deux fautes) ne l'étaient pas dans le modèle de faute, plus général que celui des auteurs de ces contre-mesures, que j'avais défini. La troisième question était alors de concevoir une nouvelle contre-mesure d'ordre supérieur, capable de résister à un nombre arbitraire de fautes. À son tour, la conception d'une nouvelle contre-mesure soulève la question de ce qui fait réellement fonctionner une contre-mesure. Pour tenter de répondre à cette question, j'ai classifié les contre-mesures existantes en essayant d'extraire les principes de protection des techniques employées. Ce travail de catégorisation m'a permis de comprendre l'essence d'une contre-mesure, et, en me basant dessus, de proposer une recette de conception de contre-mesure pouvant résister à un nombre arbitraire (mais fixé) de fautes.
→ Les résultats de ce travail ont été présentés et publiés dans la conférence internationale avec comité de lecture FDTC 2014 (à Busan, en Corée) [RG14c].

J'ai aussi remarqué que toutes les contre-mesures que j'ai étudiées sont des variantes d'optimisation d'une même technique de base qui consiste à vérifier l'intégrité du calcul en utilisant une forme de redondance homomorphe. Cette technique est indépendante de l'algorithme auquel elle s'applique, et aussi de la condition de succès d'attaque, puisqu'elle repose entièrement sur des propriétés des structures mathématiques dans lesquelles se trouve les données sensibles, c'est à dire l'arithmétique modulaire. J'ai donc proposé une propriété de résistance face aux attaques par injection de faute qui dépasse la notion de condition de succès d'attaque. La quatrième question a été d'appliquer cette technique de protection à tous les calculs de cryptographie asymétrique, puisqu'ils travaillent tous sur des données mathématiques similairement structurées. Dans cette optique, j'ai développé une abstraction des calculs de cryptographie asymétrique qui permet d'appliquer simplement la méthode de protection par redondance. J'ai formellement défini cette méthode en définissant une transformation de code par réécriture que j'ai prouvée correcte. J'ai écrit un compilateur (enredo) qui automatise cette transformation et a permis d'obtenir des implémentations protégées d'algorithmes pour lesquels aucune contre-mesure n'a été publiées mais qui sont déjà victimes de nombreuses attaques par injections de fautes, comme les calculs de multiplications scalaire sur courbe elliptiques ou les algorithmes de couplage. Un avantage additionnel du compilateur enredo est de permettre d'étudier le compromis entre sécurité et temps de calcul.
→ Les résultats de ce travail ont été rédigés dans un article [RGM+] ainsi que dans un chapitre d'ouvrage [RMG16].

Post-doctorat

Formal Model for Privacy as Control

Plutôt que le "droit d'être laissé tranquille", comme originellement défini par Samuel Warren et Louis Brandeis, le respect et la protection de la vie privée (*privacy*) est de plus en plus vu, dans la société du tout numérique, comme le contrôle que peut exercer un individu sur ses données personnelles. La mode est d'ailleurs à l'inclusion d'exigences de respect et de protection de la vie privée dès les premières phases de conception d'un produit ou service, en suivant l'approche "*privacy by design*". Cependant, alors même que les notions de "*privacy as control*" et "*privacy by design*" sont omniprésentes dans la littérature, aucune définition claire de leur signification n'existe à ce jour. En conséquence ces notions peuvent être interprétées de différentes manières et il est difficile de rendre leur mise en pratique systématique ou mesurable. Par exemple, la littérature informatique en matière de vie privée se concentrent principalement autour de deux sujets : les "policy languages" qui permettent d'exprimer des politiques de gestions des données personnelles, et le contrôle d'accès qui a développé des variantes comme les "role-based access control", le "purpose-based access control", ou encore le "risk-adaptive access control" pour mettre en œuvre la notion de "privacy as control", sans jamais vraiment définir ce que c'est réellement.

Il ressort cependant de la littérature, des pratiques, et de la loi (vue au travers des recommandations de mise en application données par la CNIL), que le contrôle d'un individu sur ses données personnelles s'organise autour de

trois axes : l'*utilisation* de ses données personnelles, le *consentement* à l'utilisation de ses données personnelles par d'autres, et la *connaissance* qu'il·le a de l'utilisation qui est faites de ses données personnelles.

Je travaille donc en ce moment à la mise au point d'un modèle formel du contrôle se basant sur ces trois axes. Plus précisément, je suis en train de concevoir un langage de modélisation du contrôle des utilisateurs sur leurs données personnelles dans un système d'information. Le but étant de créer un outil qui prendrait en entrée la description d'un système dans ce langage, ainsi qu'un ensemble de critères définissant ce que serait un "contrôle satisfaisant", et procéderait à une vérification automatique de la validité de ces critères dans tous les états du système décrit. Le cas échéant, l'outil permettrait de tracer d'où vient la perte de contrôle pour pouvoir y pallier, ou du moins émettre des recommandations d'amélioration du système étudié, que ce soit au niveau de ses spécifications ou de moyen de contrôles externes.

Ce modèle formel et l'outil qui l'implémente, pourront servir de socle à la création d'une mesure du niveau de respect et de protection de la vie privée qu'offre un système donné, et donc de quantifier le risque de perte de contrôle, ou encore d'évaluer l'efficacité de méthode de protection du contrôle. À son tour, cette mesure devrait permettre d'extraire des principes généraux de bonnes pratiques vis-à-vis du contrôle des utilisateurs sur leurs données personnelles, et donc de définir plus précisément la notion de "privacy by design" afin de rendre plus facile l'implémentation de produits ou services respectueux et protecteur de la vie privée.

Ce projet venant juste de démarrer (octobre 2015) il n'y a pas encore de résultat publié.

Collaborations internationales

Pendant ma thèse je suis allé travailler un mois à Madrid avec Gilles Barthe dans son groupe **Computer-Assisted Cryptography à l'IMDEA Software Institute**. J'ai collaboré avec eux sur le démarrage de mon travail de formalisation et d'automatisation (enredo) de la réécriture de code de calculs arithmétiques (type cryptographie asymétrique) pour la protection automatique contre les attaques par injections de fautes. C'est à la suite de ce travail que j'ai encadré moi-même un stagiaire de M2 (Martin Moreau, qui est maintenant en thèse avec Gilles Barthe à l'IMDEA) pour poursuivre cette idée et la mettre en pratique, ce qui a donné naissance à la rédaction d'un article [RGM+], ainsi que d'un chapitre d'ouvrage [RMG16].

J'ai aussi collaboré pendant ma thèse avec Ágnes Kiss et Juliane Krämer de la Technische Universität Darmstadt en Allemagne, à la rédaction d'un article [KKR+] faisant suite à un travail qu'elles ont effectué en utilisant une des méthodes que j'ai développées pendant ma thèse ainsi que l'outil (finja) que j'ai écrit pour l'implémenter.

Dissémination

Publications : revues internationales avec comité de lecture (×2), conférences internationales avec comité de lecture (×7), conférences internationales sans comité de lecture (×3), chapitre d'ouvrage (×1).
Viennent s'ajouter 3 articles en cours de préparation.

Exposés : conférences internationales avec comité de lecture (×6), conférence nationale (×1), séminaires de recherche (×4 dont 2 invités, sans compter les présentations dans mon propre laboratoire), sessions posters de conférences internationales (×2).
Les présentations est posters se trouvent sur ma page web : `http://pablo.rauzy.name/research.html`.

Projets

Enseignement

J'aimerais beaucoup enseigner les bases fondamentales de l'informatique en licence : mathématiques discrètes, structures de données, logique, automates, fonctions récursives, lambda calcul, machine de Turing, etc.
Faire des cours sur des langages de programmation (à tous niveaux) m'intéresse aussi : faire un cours de C, enseigner un langage fonctionnel (Scheme ou OCaml par exemple), ou encore des paradigmes de programmation moins courants comme la programmation synchrone (Lustre), ou la programmation logique (Prolog).
Je suis aussi motivé à l'idée d'enseigner la programmation système (POSIX, pas Windows) et réseau bas-niveau.
Enfin, faire des cours sur des sujets orienté sécurité, en particulier en lien avec mes recherches, me plairait aussi énormément bien sûr, mais plutôt en niveau master.

Recherche

Mes intérêts scientifiques se portent sur la *sécurité de l'information*, notamment dans le but du respect et de la protection de la *vie privée*. Ce domaine m'intéresse à tous les niveaux, de la sécurité physique des implémentations de systèmes embarqués jusqu'à l'ingénieurie sociale, en passant par la cryptologie, la sécurité des systèmes et réseaux, la rétroingénieurie, l'analyse de vulnérabilité et de malware, et le test d'intrusion.
Je crois à la fois fermement en l'efficacité de l'*approche offensive* de la sécurité et en la nécessité de l'*approche formelle* pour la preuve et l'automatisation des méthodes de protection des systèmes.
Mon projet est de développer de nouveaux modèles formels permettant d'une part la mise en œuvre de automatisée de découverte de vulnérabilité, et d'autre part la preuve de contre-mesure et si possible l'automatisation de la protection.

Pablo Rauzy

✉ pablo.rauzy@inria.fr – r@uzy.me
http://pablo.rauzy.name/
26 ans, français

Publications et logiciels – *Qualification aux fonctions de Maître de Conf.*

Des ⋆ marquent les trois articles principaux qui sont inclus dans mon dossier de candidature. Les articles pour lesquels les noms de auteurs sont classés par ordre alphabétique plutôt que par ordre de contributions sont marqués d'un α. Toutes mes publications sont disponibles en version intégrale depuis ma page web : `http://pablo.rauzy.name/research.html#publications`

Revues internationales avec comité de lecture

[RGN15] Pablo Rauzy, Sylvain Guilley, Zakaria Najm. **Formally Proved Security of Assembly Code Against Power Analysis**. *Journal of Cryptographic Engineering*, issue à paraître, 2015. DOI: 10.1007/s13389-015-0105-2. ⋆

[RG14a] Pablo Rauzy, Sylvain Guilley. **A Formal Proof of Countermeasures Against Fault Injection Attacks on CRT-RSA**. *Journal of Cryptographic Engineering*, Volume 4 Issue 3, 2014. DOI: 10.1007/s13389-013-0065-3.

Chapitres d'ouvrage

[RMG16] Pablo Rauzy, Martin Moreau, Sylvain Guilley. **Protecting Pairings-Based Cyrptography Against Fault Injection Attacks**. Chapitre du livre *Handbook of Pairing Based Cryptography*, Nadia El Mrabet et Marc Joye éditeurs, CRC Press Taylor and Francis group, à paraître.

Conférences internationales avec comité de lecture

[RNR+15] Lionel Rivière, Zakaria Najm, Pablo Rauzy, Jean-Luc Danger, Julien Bringer, Laurent Sauvage. **High Precision Fault Injections on the Instruction Cache of ARMv7-M Architectures**. *HOST 2015: IEEE International Symposium on Hardware-Oriented Security and Trust*. DOI: 10.1109/HST.2015.7140238.

[RG14c] Pablo Rauzy, Sylvain Guilley. **Countermeasures Against High-Order Fault-Injection Attacks on CRT-RSA**. *FDTC 2014: 11th IACR Workshop on Fault Diagnosis and Tolerance in Cryptography*. DOI: 10.1109/FDTC.2014.17. ⋆

[RG14b] Pablo Rauzy, Sylvain Guilley. **Formal Analysis of CRT-RSA Vigilant's Countermeasure Against the BellCoRe Attack**. *PPREW 2014: 3rd SIGPLAN Program Protection and Reverse Engineering Workshop*. DOI: 10.1145/2556464.2556466.

[ABB+12] Antoine Amarilli, Fabrice Ben Hamouda, Florian Bourse, Robin Morisset, David Naccache, Pablo Rauzy. **From Rational Number Reconstruction to Set Reconciliation and File Synchronization**. *TGC 2012: 7th International Symposium on Trustworthy Global Computing*. DOI: 10.1007/978-3-642-41157-1_1. α (article invité)

[ANR+11] Antoine Amarilli, David Naccache, Pablo Rauzy, Emil Simion. **Can a Program Reverse-Engineer Itself?**. *IMACC 2011: 13th IMA International Conference on Cryptography and Coding*. DOI: 10.1007/978-3-642-25516-8_1. α (article invité)

[AMN+11] Antoine Amarilli, Sascha Müller, David Naccache, Daniel Page, Pablo Rauzy, Michael Tunstall. **Can Code Polymorphism Limit Information Leakage?**. *WISTP 2011: Workshop in Information Security Theory and Practice*. DOI: 10.1007/978-3-642-21040-2_1. α (article invité)

Les articles [AMN+11], [ANR+11], et [ABB+12] correspondent à des travaux effectués avant ma thèse.

Conférences internationales avec comité de lecture mais sans actes

[RG15] Pablo Rauzy, Sylvain Guilley. **Towards Generic Countermeasures Against Fault Injection Attacks**. *TRUDEVICE 2015: 3rd Workshop on Trustworthy Manufacturing and Utilization of Secure Devices*.

[RGN14] Pablo Rauzy, Sylvain Guilley, Zakaria Najm. **Formally Proved Security of Assembly Code Against Power Analysis**. *PROOFS 2014: 3rd Workshop on Security Proofs for Embedded Systems*. Sélectionné pour soumission en version étendue au *Journal of Cryptographic Engineering*.

[RG13] Pablo Rauzy, Sylvain Guilley. **A Formal Proof of Countermeasures Against Fault Injection Attacks on CRT-RSA**. *PROOFS 2013: 2nd Workshop on Security Proofs for Embedded Systems*. Sélectionné pour soumission en version étendue au *Journal of Cryptographic Engineering*.

[RGD13] Pablo Rauzy, Sylvain Guilley, Jean-Luc Danger. **Software Countermeasures Against DPA Attacks: Masking vs Dual-Rail with Precharge Logic**. *COSADE 2013: 4th International Workshop on Constructive Side-Channel Analysis and Secure Design*. (article court)

Articles en préparation

[RL] Pablo Rauzy, Daniel Le Métayer. **Modeling Privacy as Control**.

[RGM+] Pablo Rauzy, Sylvain Guilley, Martin Moreau, Zakaria Najm. **Using Modular Extension to Provably Protect ECC Against Fault Attacks**. ⋆

[KKR+] Ágnes Kiss, Juliane Krämer, Pablo Rauzy, Jean-Pierre Seifert. **Algorithmic Countermeasures Against Fault Attacks and Power Analysis for RSA-CRT**.

Logiciels

paioli Cet outil permet de protéger du code assembleur contre les attaques par analyse de consommation de courant (comme la DPA ou la CPA) et de formellement prouver l'efficacité de la protection. Pour cela, il implémente l'insertion automatique d'une contre-mesure d'équilibrage connu sous le nom de DPL (*dual-rail with precharge logic*) dans du code assembleur "bitslicé" (typiquement sur un algorithme de chiffrement par bloc). Indépendemment, il est capable, en exécutant symboliquement le code un peu à la manière de l'interprétation abstraite, de vérifier statiquement si la consommation de courant d'un code assembleur donné est correctement équilibré au regard d'une modèle de fuite (typiquement la distance de Hamming des mises à jour de valeurs, et le poids de Hamming des valeurs).

La création de cet outil et les résultats obtenus avec ont été publiés [RGN14,RGN15].

De plus, une version protégée de l'algorithme de chiffrement PRESENT qui a été obtenue avec paioli est actuellement utilisée et étudiée dans le groupe Microsystems à la Nanyang Technological University de Singapoure.

Développé à 100% par moi. Code source et exemples disponibles sous licence CeCILL sur `http://pablo.rauzy.name/sensi/paioli.html`.

finja Cet outil permet l'analyse formelle de contre-mesure contre les attaques par injection de fautes de type BellCoRe sur les algorithmes de cryptographie asymétrique du type de CRT-RSA. Il utilise les règles de l'arithmétique modulaire pour faire de l'évaluation symbolique par réécriture, ce qui permet une couverture complète des fautes possibles dans un modèle d'attaque donné (fautes aléatoires et/ou à zéro, permanente et/ou transiente, nombre de fautes) afin de prouver la résistance d'une contre-mesure en fonction d'une condition de succès d'attaque.

La création de cet outil et les résultats obtenus avec ont donnés lieu à trois publications [RG14a,RG14b,RG14c]. L'outil et les résultats qu'il a permis ont été apprécié de la communauté scientifique, par exemple pour la simplification des codes et la diminutions de la quantité de nombres aléatoires (coûteux à générer) nécessaires dans les contre-mesures de l'état de l'art, comme celles de Aumullër et al. (Infineon) et de Vigilant (Gemalto).

Cet outil a été réutilisé par Ágnes Kiss, Juliane Krämer, et Jean-Pierre Seifert de TU Darmstadt en Allemagne pour étudier formellement d'autres types de contre-mesures pour CRT-RSA que celles que j'ai étudié moi. Leur travail a abouti à un article sur lequel nous sommes en train de collaborer [KKR+].

Développé à 100% par moi. Code source et exemples disponibles sous licence CeCILL sur `http://pablo.rauzy.name/sensi/finja.html`.

enredo Cet outil permet de protéger les algorithmes de cryptographie asymétrique contre les attaques par injection de faute. Il applique une transformation de code (sur du pseudo-code, mais peut sortir du Python ce qui permet d'effectuer des tests) prouvée correcte, qui insère la contre-mesure prouvée correcte appelée *extension modulaire* qui permet à faible coût d'introduire une redondance dans le calcul, et ainsi de vérifier son intégrité.

Contrairement aux deux précédents, la principale description de cet outil se trouve uniquement dans ma thèse (chapitre `0x8`). Un article se basant sur les résultats de cet outil est en cours de préparation [RGM+], et un chapitre d'ouvrage [RMG16] va paraître.

L'écriture de cet outil a été démarré lors de mon stage à l'IMDEA Software Institute avec Gilles Barthe.

Développé à 100% par moi. Code source et exemples disponibles sous licence CeCILL sur `http://pablo.rauzy.name/sensi/enredo.html`.

ATTESTATION

DOCTORAT DE TELECOM PARISTECH

Le Directeur Adjoint de l'Ecole Doctorale d'Informatique, Télécommunications et Electronique (Edite de Paris) atteste que :

Pablo RAUZY
né le 30 juillet 1989 à Marseille (France)
de nationalité française

a soutenu le 13 juillet 2015 une thèse de Doctorat intitulée :

« Méthodes logicielles formelles pour la sécurité des implémentations de systèmes cryptographiques»

Le jury, présidé par le Professeur David NACCACHE , après délibérations, lui a attribué le grade de Docteur de ***Télécom ParisTech***, dans la spécialité « **Informatique et Réseaux** ».

Fait à Paris le20/11/2015, à la demande de l'intéressé pour servir et valoir ce que de droit.

Le Directeur de la Formation Doctorale

Alain SIBILLE

Institut Mines-Télécom
Télécom ParisTech
46 rue Barrault
75634 PARIS Cedex 13
FRANCE

TELECOM ParisTech

PROCES VERBAL DE SOUTENANCE

Les soussignés, constitués en jury de thèse, conformément à l'arrêté du 7 août 2006 relatif aux études doctorales,

Président :	DAVID NACCACHE	
Rapporteurs :	Pierre-Alain FOUQUE	Professeur Université de Rennes 1, Rennes
	Marie-Laure POTET	Professeur Grenoble INP Ensimag, Gières
Examinateurs :	François DUPRESSOIR	Post-Doctoral Researcher IMDEA Software Institute, Madrid, Espagne
	David NACCACHE	Professeur Université de Paris II, Paris
	Karine HEYDEMANN	Maître de Conférences Université Pierre et Marie Curie, Paris
	Medhi TIBOUCHI	Researcher NTT Secure Platform Laboratories, Tokyo, Japon
Directeurs de thèse :	Sylvain GUILLEY	Professeur Télécom ParisTech, Paris
	Jean-Luc DANGER	Directeur d'Études Télécom ParisTech, Paris

ont examiné le mémoire de thèse de doctorat de

Pablo RAUZY
né le 30 juillet 1989 à Marseille (France)
de nationalité française

Intitulée **«Méthodes logicielles formelles pour la sécurité des implémentations de systèmes cryptographiques»**

La présentation orale a conduit à la rédaction du rapport suivant :

Le candidat a présenté ses travaux concernant la protection et l'attaque d'une grande variété d'algorithmes cryptographiques, symétriques et asymétriques, face à eux catégories d'attaques : les attaques par fautes et les attaques par canaux cachés.

Le jury souligne la rigueur, la très grande originalité et l'aspect novateur des travaux du candidat. Ces contributions, majeures et à très large couverture, lancent le nouveau domaine de recherche que constitue l'application d'outils formels hautement automatisés à l'analyse d'algorithmes d'implémentations cryptographiques. Les résultats fournissent des garanties de sécurité essentielles au déploiement de la cryptographie embarquée.

Les travaux du candidat démontrent une excellente maitrise des mathématiques, statistiques, cryptologie, électronique et informatique. En particulier, le jury souligne que les compétences informatiques du candidat couvrent un très large spectre allant de l'assembleur aux langages formels de haut niveau. Il s'agit d'un spectre de connaissances exceptionnellement large qui ferait du candidat un excellent enseignant-chercheur ou un excellent chercheur.

L'exposé a été d'une très grande clarté et d'une très grande pédagogie. Les réponses apportées aux nombreuses questions du jury ont été pertinentes et précises.

Ainsi, le jury déclare M. Pablo Rauzy docteur en informatique de l'Institut Mines-Télécom ParisTech avec la mention « Très Honorable ».

Après délibération, les soussignés l'ont déclaré digne du grade de Docteur de Télécom ParisTech.

dans la spécialité : Informatique et Réseaux

avec la mention : ..... TRÈS HONORABLE .....

Fait à Paris, le 13 juillet 2015

Le Président :

Les Rapporteurs :

Les Examinateurs :

TIBOUCHI

TELECOM ParisTech

AVIS DU JURY SUR LA REPRODUCTION DE LA THESE SOUTENUE

«Méthodes logicielles formelles pour la sécurité des implémentations de systèmes cryptographiques»

Spécialité : Informatique et Réseaux

Prénom et nom de l'auteur : **M. Pablo RAUZY**

Président du jury : DAVID NACCACHE

Date de la soutenance : le 13 juillet 2015

Reproduction de la thèse soutenue [1] :

- ☒ Thèse pouvant être reproduite en l'état.
- ☐ Thèse pouvant être reproduite après corrections suggérées au cours de la soutenance.
- ☐ Thèse devant faire l'objet d'une diffusion restreinte ou différée pour des contraintes de confidentialité ou de propriété industrielle.
- ☐ Thèse ne pouvant être reproduite, à soumettre au président du jury après modifications (ci-jointes)

Signature du président du jury :

1) Cocher une des cases

Rapport sur le manuscrit
"Formal Software Methods for Cryptosystems Implementation Security"
présenté par Pablo Rauzy pour l'obtention
du doctorat de Telecom ParisTech

Pierre-Alain Fouque

22 juin 2015

Les attaques par canaux auxiliaires sont des attaques très efficaces contre les implémentations d'algorithmes cryptographiques. Elles permettent en temps raisonnable de retrouver des éléments secrets en exploitant des informations complémentaires sur des variables internes du calcul. En cryptographie classique, l'adversaire ne peut pas avoir accès à ces informations, mais ce type d'attaque est réalisable quand les composants cryptographiques sont embarqués dans de petits dispositifs. Dans certains cas, il est possible d'injecter une erreur pendant un calcul qui s'effectue dans un composant électronique ou un processeur embarqué, en utilisant un rayonnement électromagnétique ou un laser. L'exploitation de la sortie de l'algorithme fautée et non fautée permet bien souvent de retrouver la clé secrète. C'est le cas pour l'algorithme de calcul rapide de la fonction RSA qui utilise le théorème des restes chinois (CRT) et qui permet de gagner un facteur environ 4 dans le temps de calcul.

Dans ce manuscrit Pablo Rauzy présente des contributions originales à la recherche d'attaque par fautes contre les implémentations cryptographiques de schémas à clé publique et en particulier les implémentations de l'algorithme RSA-CRT. Il décrit aussi comment se protéger contre ces attaques. Enfin, un outil pour diminuer l'information qui fuit d'une implémentation est présenté. Les techniques utilisées dans cette thèse sont issus des méthodes formelles afin d'étudier directement les implémentations cryptographiques de façon exhaustive.

Travaux

Dans le premier chapitre de cette thèse, Pablo Rauzy décrit un outil appelé paioli destiné à compiler du code assembleur en code assembleur plus sûr contre les attaques par canaux auxiliaires par analyse de courant. Ces attaques exploitent la variation de consommation de puissance quand un transistor passe de l'état 0 à 1. Ainsi, en enregistrant cette variation physique, il est possible d'obtenir le poids de Hamming de certaines variables du calcul. En matériel, une contremesure consiste à utiliser du

matériel qui ne fuit pas en poids de Hamming. Il est possible de coder par exemple l'état `0` avec deux bits `(01)` et l'état `1` avec `(10)` de sorte que la consommation soit constante. L'idée de l'outil paioli consiste à simuler de façon logicielle ces mécanismes afin de diminuer l'information obtenue par l'adversaire. Les expérimentations sont très convainquantes sur un processeur bien connu pour fuir selon ce modèle. L'outil permet de compiler du code assembleur vers une implémentation plus sûre.

Plusieurs chapitres qui suivent dans le manuscrit de Pablo étudient la sécurité des attaques par fautes contre l'algorithme RSA-CRT. Pablo a développé l'outil finja qui étudie de façon exhaustive les fautes randoms et qui mettent une variable à zéro dans plusieurs implémentations de l'algorithme haut-niveau de l'algorithme RSA-CRT (c'est-à-dire qu'il ne rentre pas dans le détail des appels de fonction d'exponentiation modulaires ou de multiplications modulaires). Cet outil permet d'analyser la sécurité de nombreuses contre-mesures. Les résultats sont très remarquable et en particulier, Pablo a été capable de retrouver automatiquement certaines attaques, d'en découvrir de nouvelles et de proposer des contremesures plus efficaces car certains tests ont été enlevés car il ne servait à rien. Les tests sont une parade efficace pour lutter contre les attaques par fautes et Pablo a analysé les contremesures proposées par Shamir, Ausmuller et al., Vigilant et Coron et al.. L'outil effectue une recherche exhaustive sur les différentes variables d'un calcul, propage ces contraintes et vérifie si une condition de faute qui mène à une attaque est vérifiée. Les attaques recherchées sont du type Bellcore Attack et permettent de retrouver les facteurs secrets du module RSA avec un simple calcul de pgcd. Les résultats sont impressionnants car dans ce domaine les attaques étaient découvertes à la main et de nombreuses erreurs ont été commises dans les contremesures proposées.

Enfin, Pablo s'est intéressé dans ce manuscrit à proposer une contremesure contre les attaques par fautes pour la cryptographie à clé publique. La contremesure est dans la suite des contremesures proposées pour sécuriser l'algorithme RSA-CRT en utilisant de la redondance. Les calculs ne sont pas seulement effectués modulo p et q, mais modulo pr et qr avec r un petit entier, ce qui permet de vérifier modulo r si une faute est introduite. Pablo a généralisé ce mécanisme pour les calculs sur un corps fini très utile en cryptographie qui utilise les courbes elliptiques par exemple. Il a développé un compilateur qui prend en entrée une implémentation et retourne un code protége en incluant la contremesure de façon automatique. Ensuite, plusieurs tests ont été fait afin de vérifier expérimentalement la résistance des implémentations. Ce travail est très prometteur et démontre la puissance des techniques développées par Pablo Rauzy.

Contributions

Les contributions de Pablo Rauzy sont nombreuses et ont été publiées dans des conférences internationales importantes dans son domaine de recherche :

1. l'outil paioli qui permet de protéger les implémentations contre les attaques passives par canaux auxiliaires en émulant les techniques développées dans le

protocole Dual-rail with Precharge Logic (DPL) pour équilibrer en consommation la fuite d'information et qui a été publié dans la conférence internationale PROOFS 2014 et au journal of Cryptographic Engineering (JoCE).

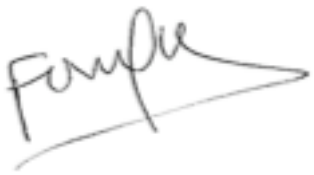
2. L'outil finja qui a été développé pour rechercher des attaques par fautes contre certaines implémentations de l'algorithme RSA-CRT et présenté à PROOFS 2013 et une extension à PPREW 2014.
3. Des contremesures ont été présentées et analysées à FDTC 2014 et en soumission journal actuellement.
4. L'outil enredo de contremesure universelle et générique pour protéger les implémentations de cryptographie à clé publique en utilisant la technique dite de l'entanglement. Ce travail est en soumission actuellement.

D'autres contributions n'ont pas été intégrées dans le manuscrit mais témoigne de son activité de recherche multiple avant thèse et actuelle.

Conclusion

Le manuscrit de Pablo Rauzy présente de nombreuses analyses à la fois en conception et en cryptanalyses très intéressantes en exploitant la puissance des méthodes formelles quand les contremesures de l'algorithme RSA-CRT utilisent de plus en plus de tests et deviennent complexes.

En conclusion, il s'agit d'une excellente thèse qui étudie la sécurité des implémentations de l'algorithme RSA-CRT en développant trois outils utilent aux concepteurs. Il est exceptionnel que les étudiants développent autant d'outil de ce niveau dans une thèse. Par conséquent, je donne un avis très favorable à la soutenance.



Pierre-Alain Fouque

Verimag
Unité mixte de recherche
Centre Equation

Grenoble INP ENSIMAG

Gières le 19 juin 2015

Rapport sur le mémoire de thèse intitulé « Méthodes logicielles formelles pour la sécurité des implémentations cryptographiques » présenté par Pablo Rauzy

Le développement de techniques de durcissement d'applications sécurisées, contre différents types d'attaques (vulnérabilités, canaux cachés, perturbations physiques), est un domaine de recherche largement ouvert, en raison de la nécessité de certifier de plus en plus d'applications dans des contextes de plus en plus ouverts, et aussi de par l'évolution des techniques d'attaques. Par exemple l'état de l'art permet maintenant des injections de fautes multiples, ce qui nécessite de revisiter les algorithmes considérés comme robustes jusqu'à présent et introduit une combinatoire montrant les limites des approches basées sur l'expertise ou l'évaluation manuelle.

Il devient donc nécessaire de mettre en place des processus d'analyse reproductibles et adaptables à différents modèles de fautes. D'autre part, même si les algorithmes sont prouvés robustes, il devient aussi nécessaire de s'intéresser aux implémentations qui peuvent, elles-aussi, introduire des structures vulnérables à l'injection de fautes (par exemple faire « sauter » une contre-mesure).

Le travail présenté par Pablo Rauzy dans ce manuscrit s'inscrit dans cette problématique, en proposant des outils et une méthodologie, basés sur l'utilisation de méthodes formelles, pour mener des vérifications à la fois sur des modèles mais aussi sur des implémentations, dans le cadre d'attaques par canaux cachés ou par injection de fautes. L'outillage proposé est appliquée sur des algorithmes de chiffrement asymétriques implémentant différentes contre-mesures (par exemple CRT-RSA avec les différentes contre-mesures proposées par Shamir, Aumüller et all ou Vigilant). Ce travail est novateur, dans le sens où il établit un pont entre la démarche des concepteurs d'algorithmes cryptographiques robustes et la démarche des informaticiens, se basant sur des langages et des outils effectifs d'analyse pour ces langages permettant l'automatisation, la reproductivité et l'adaptation.

Le manuscrit est constitué de 180 pages (incluant les annexes) et est composé de 9 chapitres. Les chapitres correspondent en général à des publications et peuvent être vus comme auto-contenus. En premier lecture ceci surprend mais le fil directeur est bien présent. Le style est direct et très agréable : les notions nécessaires sont en général clairement introduites et la démarche adoptée dans le manuscrit est argumentée et justifiée tout du long du manuscrit. L'auteur n'hésite pas à donner des points de vue personnels et étayés, qui bénéficient du double point de vue implémentation cryptographique et génie logiciel.

Le premier chapitre (numéroté 3) balaie rapidement toutes les notions utiles à ce travail et l'état de l'art : des attaques et contre-mesures aux méthodes formelles. Il introduit le contenu détaillé des chapitres et les contributions de la thèse.

Le chapitre 4 constitue un des premiers apports de ce travail, qui se concrétise par la réalisation d'un outil (Paioli) dont l'objectif est d'appliquer des modifications de code bas niveau pour assurer la robustesse à la fuite d'information par analyse de consommation type DPA et CPA. L'auteur propose une implémentation logicielle de la contre-mesure classique « Dual-Rail with Precharge Logic » par une transformation de code, prouvée correcte. On aurait aimé en savoir plus sur son fonctionnement et sur les limitations inhérentes à l'utilisation de cette technique (obtenir un graphe de flot, énumérer les chemins et résister à des formules complexes).

Le chapitre 5 démarre par une étude générale des contre-mesures des implémentations RSA contre des attaques du type BellCoRe. La contribution ici est l'outil finja qui permet d'analyser la robustesse de telles implémentations en se basant sur un système de réécriture symbolique pour l'arithmétique modulaire. La méthode proposée est expérimentée sur 3 implémentations : une implémentation naïve, une implémentant la contre-mesure de Shamir et la dernière celle de Aumüller et all. Cette dernière est établie correcte vis-à-vis du modèle de fautes considéré. Le chapitre 6 étend l'approche et l'outil finja à la contre-mesure de Vigilant et la version proposée par Coron et al. L'approche proposée dans ces deux chapitres constitue une contribution importante au domaine, de par l'aspect généralisation de l'analyse des contre-mesures et l'outil proposé.

Le chapitre 7 aborde le problème important de la caractérisation des contre-mesures, leur classification et leur évaluation vis-à-vis d'un modèle d'attaquant, en particulier dans le cadre des fautes d'ordre supérieure (plusieurs injections possibles). Une transformation entre les contre-mesures dites « test-based » et « infective » est donnée et permet de comparer différentes solutions en termes de protection. Cette étude est utilisée, en lien avec l'outil finja, pour optimiser le nombre de vérifications dans la contre-mesure de Vigilant et pour proposer une méthode systématique de durcissement contre des fautes multiples, partant d'une contre-mesure correcte pour une attaque en une faute.

Le dernier chapitre étend les résultats et analyses précédentes en généralisant les contre-mesures par extension modulaire pour des anneaux finis dans le cadre de la cryptographie asymétrique. Un ensemble de transformations, implémentées dans l'outil enredo, permet d'implémenter cette contre-mesure. Une preuve, non triviale, de la sécurité de la transformation proposée. Ce chapitre finit par une expérimentation qui valide en pratique les transformations proposées.

En conclusion, le travail proposé dans cette thèse constitue une avancée très intéressante dans le domaine du durcissement d'applications sécurisées contre les attaques par canaux cachés et par injection de fautes, appliquée ici à la cryptographie asymétrique. L'utilisation des méthodes formelles telle que proposée par Pablo Rauzy permet de mettre en place une véritable démarche prouvée d'analyse de robustesse et des contre-mesures, et de mécanisation de ces analyses. Dans ce travail, Pablo Rauzy a montré sa parfaite maîtrise du domaine des implémentations cryptographiques basées sur RSA et de l'état de l'art en terme de contre-mesures, ainsi qu'une grande maturité lui permettant de proposer une démarche et des outils pertinents. On aurait néanmoins aimé en savoir un peu plus sur la réalisation des outils, leurs limitations et l'applicabilité à d'autres types de code (cryptographique ou non). Comme tout travail un tant soit peu novateur, le travail de thèse de Pablo Rauzy lève de nombreuses questions et ouvre des perspectives qui ont déjà commencées à être explorées par de nouveaux travaux.

Pour toutes ces raisons, je suis très favorable à la soutenance de ce travail par Pablo Rauzy pour obtenir le grade de docteur, spécialité informatique, de TELECOM ParisTech.

Marie-Laure Potet
Professeur des Universités, Grenoble INP
Laboratoire Vérimag

Sylvain Guilley.
Professeur au département
COMELEC de TELECOM-ParisTech,
Institut Mines-Télécom,
Laboratoire LTCI (UMR 5141).

Objet: Avis sur la thèse de Pablo Rauzy

J'ai eu le plaisir de rencontrer Pablo Rauzy en 2012 via l'entremise de David Naccache. J'ai dirigé ses travaux de thèse de doctorat, qui ont porté sur le rôle des méthodes formelles dans le domaine de la sécurité de l'information. Plus précisément, pendant sa thèse, Pablo a travaillé à renforcer le niveau de confiance des protections mises en œuvre pour augmenter la robustesse des implémentations de systèmes cryptographiques. Par exemple, les révélations récurrentes sur les failles de sécurité dans la bibliothèque OpenSSL (biais dans la génération d'aléa, *timing attacks*, *heartbleed*, etc.) montrent à quel point il est important de soigner l'implémentation des codes cryptographiques.

Pablo a obtenu des résultats innovants en matière de protections de codes cryptographiques, aussi bien contre les attaques passives (mesure de fuite via des canaux cachés) qu'actives (observation de la réaction à une perturbation). Les deux types d'attaques sont très efficaces en pratique, et sont d'ailleurs réalisées au quotidien au département COMELEC de TELECOM-ParisTech. En deux mots, Pablo a démontré la correction d'une classe d'implémentations à temps de calcul et à consommation d'énergie constants, grâce à une méthode d'évaluation symbolique, définie et spécifiée par ses soins. Dans le domaine des attaques actives, Pablo a formalisé une protection de redondance basée sur une vérification homomorphique de calculs permettant de réaliser des protocoles cryptographiques asymétriques. En plus de leur utilité pratique, ces contributions montrent qu'il est possible d'obtenir une garantie formelle de la correction d'implémentations et de la bonne mise en oeuvre de protections sur des codes logiciels ou matériels concrets (par exemple en langage assembleur).

Sans aucun doute, la thèse de Pablo (soutenue le 13 juillet 2015 à l'ENS rue d'Ulm) deviendra une référence dans le domaine novateur des méthodes formelles au service de la sécurité des implémentations cryptographiques. D'ailleurs, de nombreuses équipes (TU Berlin, Paderborn U, Université de Paris 8, etc.) sont en train de plancher sur ses résultats. De plus, Pablo a transféré lui-même son savoir-faire à un stagiaire M2 (Martin Moreau) qu'il a co-encadré.

Je suis personnellement extrêmement satisfait du travail réalisé par Pablo. Pablo est un chercheur autonome et très organisé, qui sait allier théorie à la pratique. Il a cette grande qualité d'aller au bout des choses. De plus, Pablo porte un soin

tout particulier à la clarté de ses explications. Ce souci de pédagogie se retrouve à la fois en phase de *brainstorming*, quand les concepts se créent, que lors des communications scientifiques (conférences, séminaires). Pablo est très ouvert, humainement et scientifiquement ; il a notamment réalisé certains travaux avec ses collègues de laboratoire sur des sujets connexes à sa thématique de thèse, ce qui démontre sa grande maturité scientifique. Les résultats obtenus dépassent largement ce que j'avais imaginé au début de sa thèse. Notamment, il ressort que les protections formellement prouvées et implémentables automatiquement sont possibles, et de surcroît pour tous les algorithmes utilisés aujourd'hui.

Nous avons commencé à envisager l'après-thèse avec Pablo assez tôt, ce qui a permis d'explorer diverses pistes. En plus de ses compétences en méthodes formelles et en protection de l'information, Pablo a cultivé pendant sa thèse d'autres sujets. Par exemple, il s'est intéressé à l'organisation de la recherche scientifique et à la diffusion des connaissances, au travers d'initiatives telles que des exposés à l'école normale supérieure. Il applique d'ailleurs ces idées de diffusion libre à ces propres travaux, en alliant publication scientifique et diffusion des codes sources des programmes écrits et utilisés pour illustrer les résultats. Par ailleurs, il se pose la question de la finalité de la recherche, en questionnant son utilité sociale. Un point qui lui tient particulièrement à cœur est la protection de la vie privée. En effet, les technologies de sécurité se caractérisent par une ambivalence, en ce qu'elles peuvent être utilisées, tantôt pour "faire valoir des droits" (anonymat, oubli, confidentialité, etc.), ou au contraire pour "priver l'utilisateur de ses droits" (en verrouillant une plateforme, par exemple).

Je n'ai donc nullement été surpris quand Pablo m'a fait part de son souhait de rejoindre l'équipe PRIVATICS d'INRIA pour une période post-doctorale. Je pense en effet qu'il s'intégrera parfaitement à cette équipe et à son thème de recherche. D'ailleurs, Pablo a été sélectionné pour le post-doc intitulé "*formal model for privacy as control*", qui démarrera officiellement à compter d'octobre 2015. En outre, la thématique de recherche de cette équipe s'inscrit dans le prolongement de la thèse de Pablo Rauzy. Le cursus de Pablo Rauzy est donc très cohérent, alliant mathématiques appliquées (cryptographie, statistiques), informatique théorique (compilation, méthodes formelles), et appliquées (attaques).

Je recommande donc fortement Pablo Rauzy dans sa démarche visant à candidater à un poste de maître de conférences. En particulier, je soutiens sans réserve son dossier en vue de l'obtention d'une qualification CNU en sections 26, 27, et 61.

À Paris, le 23 septembre 2015,

Sylvain Guilley

UPMC
SORBONNE UNIVERSITÉS

CONTRAT D'ENGAGEMENT
EN QUALITÉ DE DOCTORANT CONTRACTUEL

Contrat doctoral n° 100/2012

Vu le code de la recherche, et notamment son article L. 412-2 ;
Vu le décret n° 84-431 du 6 juin 1984 modifié fixant les dispositions statutaires communes applicables aux enseignants-chercheurs et portant statut particulier du corps des professeurs des universités et du corps des maîtres de conférences ;
Vu le décret n° 86-83 du 17 janvier 1986 relatif aux dispositions générales applicables aux agents non titulaires de l'État ;
Vu le décret n° 2009-464 du 23 avril 2009 relatif aux doctorants contractuels des établissements publics d'enseignement supérieur ou de recherche ;
Vu l'arrêté du 23 avril 2009 fixant le montant de la rémunération du doctorant contractuel ;
Vu la délibération du conseil d'administration de l'UPMC adoptant la charte du doctorat le 17 décembre 2007 ;
Vu la proposition du directeur de l'école doctorale :
ED130 - Informatique, télécommunications et électronique de Paris
Vu l'avis du directeur de thèse et du directeur de l'unité :
Laboratoire d'Informatique de l'école normale supérieure (LIENS)-UMR 8548
Vu la décision du conseil scientifique de l'établissement employeur (le cas échéant si l'étudiant est dans la situation prévue au 2ème alinéa de l'article 3 du décret du 23 avril 2009) ;

ENTRE LES SOUSSIGNÉS :

LE PRÉSIDENT DE L'UNIVERSITÉ PIERRE ET MARIE CURIE,
Monsieur Jean CHAMBAZ

d'une part,

ET

LE DOCTORANT CONTRACTUEL

Nom :	**RAUZY**
Prénom :	**Pablo Jean**
Date et lieu de naissance :	**30 juillet 1989** à **MARSEILLE (013 - Bouches-du-Rhône)**
Numéro de sécurité sociale :	**1 89 07 13 155 063 - 46**
Adresse :	**34 rue Maurice Arnoux 92120 MONTROUGE (FRANCE)**

Ci-après désigné « le doctorant contractuel »

d'autre part ;

Il a été convenu ce qui suit :

Article 1er : objet

Monsieur Pablo Jean RAUZY est engagé en qualité de doctorant contractuel.
L'intéressé s'engage à s'inscrire en première année de doctorat dans un établissement d'enseignement supérieur. L'employeur d'autorité pourra dans le cas contraire résilier le contrat de plein droit.

Article 2 : conditions de préparation du doctorat

Le doctorant contractuel prépare un doctorat sur le thème :

Mixing Continuous and Discrete time in a synchronous language

- Dans l'unité de recherche : **Laboratoire d'Informatique de l'école normale supérieure (LIENS)-UMR 8548** (45 rue d'Ulm 75005 Paris)
- Sous la direction de **Monsieur Marc POUZET,** directeur de thèse.

PR

UPMC
SORBONNE UNIVERSITÉS

Article 3 : durée du contrat

Le présent contrat, d'une durée de trois ans, prend effet à compter du **01/10/2012** et se termine le 30/09/2015, conformément aux dispositions de l'article 3 du décret du 23 avril 2009 susvisé.
La rupture du contrat avant son terme par l'une ou l'autre des parties s'effectue dans les conditions prévues par le titre XI du décret du 17 janvier 1986 susvisé.
Si l'inscription en doctorat n'est pas renouvelée en début d'année universitaire selon les conditions prévues par les articles 3 et 9 du décret du 13 mai 1971, il est mis fin de plein droit au contrat de doctorant contractuel au terme de la première ou de la deuxième année du contrat, dans les conditions et avec les indemnités prévues aux titres XI et XII du décret du 17 janvier 1986 susvisé.

Article 3-bis : période d'essai

Le doctorant contractuel effectue une période d'essai d'une durée de deux mois. Durant cette période, le contrat doctoral peut être rompu par le doctorant contractuel ou le Président de l'Université Pierre et Marie Curie, sans indemnité ni préavis, par lettre recommandée avec accusé de réception.

Article 4 : obligations de service

Conformément aux termes de l'article 4 du décret, les doctorants contractuels sont soumis aux dispositions générales relatives au temps de travail dans la fonction publique, telles qu'elles résultent du décret n° 2000-815 du 25 août 2000 modifié relatif à l'aménagement et à la réduction du temps de travail dans la fonction publique de l'État et dans la magistrature. Ce texte fixe à **1607 heures** le volume annuel de travail à accomplir par chaque agent.

Article 4-1 : service confié au doctorant

Le service confié au doctorant contractuel est arrêté annuellement par le Président de l'Université Pierre et Marie Curie sur proposition du directeur de l'école doctorale, après avis du directeur de thèse et du directeur de l'unité de recherche, et avis du doctorant contractuel. Ce service est fixé dans le cadre des missions définies comme suit (cocher une des cinq cases) :

☒ Le doctorant contractuel accomplira, pendant la durée de son contrat, un service annuel qui sera exclusivement consacré aux activités de recherche liées à la préparation de son doctorat, incluant le temps de formation nécessaire à la réalisation de son plan individuel de formation.

OU

Le doctorant contractuel accomplira, pendant la durée de son contrat, un service annuel qui comprendra, pour les cinq sixième de son temps de travail effectif, les activités de recherche liées à la préparation de son doctorat, et, pour un sixième de son temps de travail, une des activités parmi celles listées ci-dessous :

☐ Enseignement, pour une année reconductible tacitement deux fois, dans le cadre d'une équipe pédagogique pour un service annuel au plus égal au tiers du service annuel d'enseignement de référence des enseignants-chercheurs, défini à l'article 7 du décret du 6 juin 1984 susvisé incluant les temps de formation nécessaires à l'exercice de ses missions ;

☐ Diffusion de l'information scientifique et technique pour une durée annuelle maximale d'un sixième de son temps de travail, incluant les temps de formation nécessaires à l'exercice de ses missions ;

☐ Valorisation des résultats de la recherche scientifique et technique pour une durée annuelle maximale d'un sixième de son temps de travail, incluant les temps de formation nécessaires à l'exercice de ses missions ;

☐ Missions d'expertise effectuées dans une entreprise, une collectivité territoriale, une administration, un établissement public, une association ou une fondation pour une durée annuelle maximale d'un sixième de son temps de travail, incluant les temps de formation nécessaires à l'exercice de ses missions.

PR

Lorsque le service confié au doctorant, dans le cadre d'une des 4 missions susvisées, est effectué dans un établissement différent de l'UPMC, une convention devra être établie entre les deux établissements.

Article 4-2 : modification des missions en cours de contrat

La liste des activités fixée ci-dessus pourra être modifiée chaque année par avenant sous réserve d'obtenir l'accord conjoint du doctorant contractuel et du Président de l'Université Pierre et Marie Curie. Cet avenant précisera, notamment, la nature des missions confiées, leurs modalités d'exercice et le niveau de rémunération retenu.

Article 5 : rémunération

Le bénéficiaire du présent contrat perçoit, pour un travail à temps plein, une rémunération mensuelle brute de **1684,93 €**. Cette rémunération est indexée sur l'évolution des rémunérations de la fonction publique.
Il peut, le cas échéant, prétendre au bénéficie du supplément familial de traitement et à la prise en charge de ses frais d'abonnement de transport et de ses frais de déplacement.

Article 5-1 : cumuls

En vertu des dispositions de l'article 5 du décret n° 2009-464 du 23 avril 2009, le doctorant contractuel **ne peut cumuler son activité** avec une charge complémentaire relevant des activités susceptibles de lui être confiées dans le cadre du contrat doctoral, et en conséquence ne peut être autorisé à exercer les missions listées ci-dessus en dehors de son contrat doctoral (*cf.* art. 4-1).

Article 6 : formation

L'UPMC propose au doctorant contractuel les formations utiles à l'accomplissement des missions qui lui sont confiées. Ces formations lui seront proposées par les services compétents de l'UPMC.
La nature et le volume des formations nécessaires à l'accomplissement des missions seront inscrites dans le plan de formation de l'UPMC.

Article 7 : obligation de réserve et propriété intellectuelle

Article 7-1 : obligation de réserve et obéissance hiérarchique

Le doctorant contractuel est soumis aux obligations incombant à l'ensemble des agents publics, notamment celle d'obéissance hiérarchique et à l'obligation de réserve. Il est également tenu au secret professionnel à l'égard des tiers en ce qui concerne les activités exercées dans l'établissement.

Article 7-2 : propriété intellectuelle

Les missions confiées au doctorant au titre du présent contrat de travail comportent une mission inventive permanente.
En conséquence et conformément à la législation en vigueur en matière de propriété intellectuelle (articles L. 611-7 et R. 611-11 à R. 611-14 notamment), les inventions faites par le doctorant appartiennent à l'établissement.
Le doctorant reconnaît que l'établissement est propriétaire de tout autre résultat valorisable, protégeable ou non par un titre de propriété intellectuelle.
Ainsi, les logiciels créés par le doctorant dans le cadre du présent contrat appartiennent à l'établissement en application de l'article L.113-9 du code de la propriété intellectuelle.

En outre, le doctorant s'engage à céder à l'établissement, par le biais de cessions de droits particuliers, la propriété pleine et entière des résultats protégés par le droit d'auteur qu'il pourrait obtenir ou pourrait contribuer à obtenir.
L'établissement dispose seul du droit de déposer les titres de propriété intellectuelle correspondants aux résultats précités.
L'établissement s'engage à ce que le nom du doctorant, s'il est considéré comme inventeur, soit

PR

mentionné dans les demandes de brevets, à moins que le doctorant ne s'y oppose.
Le doctorant s'engage à donner toutes signatures et à prêter son entier concours à l'établissement pour les procédures de protection de ces résultats (notamment pour le dépôt éventuel d'une demande de brevet, son maintien en vigueur et sa défense) ainsi que pour leur exploitation et ce tant en France qu'à l'étranger.
L'ensemble de ces dispositions demeure valable à l'expiration du contrat.

Article 7-3 : confidentialité

Le doctorant s'engage à considérer comme strictement confidentielles les informations de toute nature, communiquées par tous moyens, dont il pourrait avoir connaissance à l'occasion de l'exécution du présent contrat.
Cette obligation de confidentialité reste en vigueur pendant la durée du contrat.

Article 7-4 : publications

Le doctorant doit solliciter de manière expresse de l'autorité hiérarchique, l'autorisation de publier.
Toute publication ou communication du doctorant, liée aux travaux de recherche effectués dans le cadre de ce contrat, doit explicitement mentionner le nom de l'unité de recherche et de l'établissement.
Ces dispositions demeurent en vigueur pendant la durée du contrat.

Article 8 : discipline

L'exercice du pouvoir disciplinaire s'exerce dans les conditions prévues par le titre X du décret du 17 janvier 1986 susvisé.

Article 9 : couverture sociale

Le bénéficiaire du présent contrat sera affilié au régime général de sécurité sociale pour ce qui concerne les prestations d'assurance sociales, notamment de l'assurance maladie, et au régime de l'IRCANTEC pour ce qui concerne la retraite complémentaire.
Le doctorant contractuel bénéficiera également de la législation relative aux accidents du travail et aux maladies professionnelles.

Article 10 : congés

Le doctorant contractuel bénéficie des congés prévus par les dispositions des articles 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 19 bis, 19 ter, 20, 20 bis, 21, 23, 25 et 26 du décret du 17 janvier 1986 susvisé.
Les congés annuels seront pris suivant les conditions de l'unité de recherche dans laquelle le doctorant contractuel exerce son activité de recherche mais le seront toujours pendant la durée du contrat.

Article 11 : conséquences de l'échéance du contrat

À l'issue de la période de trois ans, le doctorant contractuel cessera son activité sans que l'établissement ait à lui signifier un préavis. Il ne pourra prétendre à une indemnité de fin de contrat destinée à compenser la précarité de sa situation.
Le titulaire du présent contrat n'acquiert pas de droit à occuper ultérieurement un emploi dans l'établissement.

Article 12 : licenciement

En dehors des cas où il intervient pour raisons disciplinaires, le licenciement peut être prononcé à tout instant sous réserve de l'observation d'un préavis dont la durée est fixée par les dispositions de l'article 46 du décret du 17 janvier 1986 susvisé. Le licenciement ne peut intervenir qu'à l'issue d'un entretien préalable. La décision de licenciement est notifiée à l'intéressé par lettre recommandée avec demande d'avis de réception. Cette lettre précise le ou les motif(s) du

PR

licenciement et la date à laquelle celui-ci doit intervenir compte tenu des droits à congés annuels restant à courir et de la durée du préavis.

Article 13

Le doctorant contractuel étranger, doit être en possession d'un titre de séjour et d'une autorisation de travail en cours de validité, excepté pour les ressortissants de l'U.E. et de l'espace économique européen (E.E.E) qui fourniront une copie de leur passeport. Sont également exemptés du titre de séjour les ressortissants suisses, monégasques et andorrans. À expiration de ces documents, il s'engage à effectuer les démarches de renouvellement et à transmettre un exemplaire de ceux-ci à l'administration.

Article 14

Le titulaire du présent contrat déclare avoir pris connaissance du décret du 23 avril 2009 susvisé et annexé au présent contrat.

Fait à Paris, le 17 septembre 2012

Signature du Président
de l'Université Pierre et Marie Curie

Pour le Président de l'Université P. et M. Curie
et par délégation
Le Vice-Président Moyens et Ressources

Laurent BUISSON

Signature du doctorant contractuel
Précédée de la mention « *lu et approuvé* »

lu et approuvé

UPMC
SORBONNE UNIVERSITÉS

Direction des Ressources Humaines
Bureau des Personnels Enseignants

Programme 150 - Titre 3
Imputation budgétaire : 64111310

AVENANT N° 1 au CONTRAT N° 100/2012
D'ENGAGEMENT EN QUALITÉ DE DOCTORANT CONTRACTUEL

Vu le code de la recherche, et notamment son article L. 412-2 ;
Vu le décret n° 84-431 du 6 juin 1984 modifié fixant les dispositions statutaires communes applicables aux enseignants-chercheurs et portant statut particulier du corps des professeurs des universités et du corps des maîtres de conférences ;
Vu le décret n° 86-83 du 17 janvier 1986 relatif aux dispositions générales applicables aux agents non titulaires de l'État ;
Vu le décret n° 2009-464 du 23 avril 2009 relatif aux doctorants contractuels des établissements publics d'enseignement supérieur ou de recherche ;
Vu l'arrêté du 23 avril 2009 fixant le montant de la rémunération du doctorant contractuel ;
Vu la proposition du directeur de l'école doctorale :
ED130 - Informatique, télécommunications et électronique de Paris
Vu la décision du conseil scientifique de l'établissement employeur (le cas échéant si l'étudiant est dans la situation prévue au 2ème alinéa de l'article 3 du décret du 23 avril 2009) ;

ENTRE LES SOUSSIGNÉS :

LE PRÉSIDENT DE L'UNIVERSITÉ PIERRE ET MARIE CURIE,
Monsieur Jean CHAMBAZ

d'une part,

ET

LE DOCTORANT CONTRACTUEL

Nom : **RAUZY**
Prénom : **Pablo Jean**
Date et lieu de naissance : **30 juillet 1989** à **MARSEILLE (013 - Bouches-du-Rhône)**
Numéro de sécurité sociale : **1 89 07 13 155 063 - 46**

Ci-après désigné « le doctorant contractuel »

d'autre part ;

Il a été convenu ce qui suit :

Article 1er

L'article 4-1 du contrat initial est ainsi modifié à compter du **01/10/2012 pour une durée d'une année reconductible tacitement jusqu'au terme du contrat :**

Le doctorant contractuel accomplira, pendant la durée de son contrat, un service annuel qui comprendra, pour les cinq sixièmes de son temps de travail effectif, les activités de recherche liées à la préparation de son doctorat, et, pour un sixième de son temps de travail, une des activités parmi celles listées ci-dessous :

☒ Enseignement dans le cadre d'une équipe pédagogique, à UPMC - UFR 919 pour un service annuel au plus égal au tiers du service annuel d'enseignement de référence des enseignants-chercheurs, défini à l'article 7 du décret du 6 juin 1984 susvisé incluant les temps de formation nécessaires à l'exercice de ses missions ;

PR

- ☐ Diffusion de l'information scientifique et technique pour une durée annuelle maximale d'un sixième de son temps de travail, incluant les temps de formation nécessaires à l'exercice de ses missions ;
- ☐ Valorisation des résultats de la recherche scientifique et technique pour une durée annuelle maximale d'un sixième de son temps de travail, incluant les temps de formation nécessaires à l'exercice de ses missions ;
- ☐ Missions d'expertise effectuées dans une entreprise, une collectivité territoriale, une administration, un établissement public, une association ou une fondation pour une durée annuelle maximale d'un sixième de son temps de travail, incluant les temps de formation nécessaires à l'exercice de ses missions.

Article 2

En conséquence, les dispositions de l'article 5 dudit contrat initial, relatives à la rémunération de l'intéressé sont remplacées par les modalités suivantes :
Le bénéficiaire du présent avenant perçoit, pour un travail à temps plein, une rémunération mensuelle brute de **2.024,70 €**. Cette rémunération est indexée sur l'évolution des rémunérations de la fonction publique.
Il peut, le cas échéant, prétendre au bénéficie du supplément familial de traitement et à la prise en charge de ses frais d'abonnement de transport et de ses frais de déplacement.

Article 3

Les autres dispositions du contrat initial restent inchangées.

Fait à Paris, le 03 octobre 2012

Signature du Président
de l'Université Pierre et Marie Curie

Pour le Président de l'Université P. et M. Curie
et par délégation
Le Vice-Président délégué Ressources Humaines

Laurent EMMANUEL

Signature du doctorant contractuel
Précédée de la mention « *lu et approuvé* »

lu et approuvé

Lettre de Recommandation

Lettre de Recommandation Pablo Rauzy

Nizar Ouarti

17 Novembre 2015

Pablo Rauzy a enseigné à Polytech' UPMC plusieurs années durant lesquelles nous avons eu à interagir. Durant ces années là, j'avais pris la responsabilité du placement des moniteurs de Polytech et le retour des élèves ainsi que des professeurs a été positif. Il a notamment donné des TP pour un cours que j'animais et j'ai pu moi même être témoin de son implication. L'ensemble des retours de l'équipe pédagogique est positif, il a passé du temps pour préparer et expliquer au mieux les cours et TP qu'il a dû aborder. Je pense que Pablo à la fibre pédagogique et aime passer du temps avec les étudiants. Je pense donc qu'il ferait un très bon Maître de conférence.

Le 17 Novembre 2015,

Nizar Ouarti Mcf IPAL, UPMC

UNIVERSITÉ PIERRE ET MARIE CURIE

Laboratoire LIP-6 DECISION
4, Place Jussieu
Boîte courrier 169
75005, Paris Cedex 05

Tél. 01 44 27 86 52
Fax : 01 44 27 70 00

Paris, le 23 septembre 2015

Pablo Rauzy
56, bd Niels Bohr,
Inria - bat. CEI2,
69100, Villeurbanne

Objet : lettre de recommandation

Madame, Monsieur,

Je soussigné, Monsieur Hacène Ouzia, atteste avoir eu Monsieur Pablo Rauzy comme chargé de travaux pratiques (TPs) et travaux dirigés (TDs) au second semestre des années universitaires 2012-2013 et 2013-2014. Plus précisément, Monsieur Rauzy avait assuré des TPs et TDs de *Programmation Web* pour des élèves ingénieurs, troisième année, à l'école Polytech'Paris UPMC. Il a effectué, au total 72 heures [1].

Monsieur Rauzy a participé à cet enseignement avec beaucoup de sérieux et s'est impliqué dans les différentes tâches pédagogiques avec beaucoup de motivation.

Je vous prie d'agréer, Madame, Monsieur, mes salutations distinguées.

Hacène Ouzia

1. Equivalent TD

Laboratoire d'Informatique de Paris 6
Université Pierre et Marie Curie – CNRS (UMR 7606)

François Pêcheux
Université Pierre et Marie Curie, Laboratoire LIP6/SOC/CIAN
Tour 25-24, 4ème Bureau A420
4, Place Jussieu
75252 PARIS Cedex 05
Tel : 01 44 27 52 53
Courriel : francois.pecheux@lip6.fr

Paris, le 21/11/15

Objet : Lettre de recommandation pour M. **Pablo Rauzy**

Pour valoir ce que de droit

Madame, Monsieur,

Je, soussigné François Pêcheux, Professeur à l'UPMC, déclare connaître M. **Pablo Rauzy** depuis Septembre 2013, date de ma prise de fonction comme responsable de la matière Informatique à l'Ecole Polytechnique Universitaire (EPU) Polytech Paris UPMC.

Depuis Octobre 2012, M. Rauzy est intervenu en tant que moniteur 27ème section dans de nombreux modules informatiques dispensés à l'Ecole. Il a ainsi encadré pendant deux ans des Travaux Pratiques de programmation en C pour une formation Electronique-Informatique en alternance (3ème année), ainsi que toute une formation de programmation web aux Elèves des spécialités Agro-Alimentaire et Matériaux (3ème année). Ses deux premières années se sont très bien passées, et les élèves ont été satisfaits par sa pédagogie et son investissement. Pour toutes ces raisons, en 2014, je lui ai entièrement confié (à sa demande) un cours complet de programmation C++ à destination de la formation initiale en Electronique-Informatique, 4ème année, qui a également donné entière satisfaction.

J'ai pu me rendre compte au cours de ces années à quel point M. Rauzy est quelqu'un de sérieux, de fiable et de motivé, qui sait tirer le meilleur parti de ses connaissances et en faire profiter les élèves qu'il encadre. Il a parfaitement compris comment notre Ecole fonctionne et a toujours entretenu d'excellents rapports avec ses collègues et les responsables pédagogiques et administratifs.

Je soutiens donc sans aucune réserve sa candidature comme Maître de Conférences.

François Pêcheux
Professeur UPMC/LIP6/SoC/CIAN

UPMC
SORBONNE UNIVERSITÉS

Faculté d'Ingénierie – UFR 919
Directeur : Yves BERTHAUD

ATTESTATION DE SERVICE

Je soussigné, Yves BERTHAUD, Directeur de la Faculté d'Ingénierie, certifie que **Pablo RAUZY** affecté à la faculté d'ingénierie en qualité de chargé de mission d'enseignement de 2012 à 2015 (3 ans), a effectué les services d'enseignement suivants à Polytech Paris-UPMC :

- Année 2012-2013 : 64 TD
  - 32TD en EPU-C6-IGV– Informatique générale-MTX3-AGRAL3-S2
  - 32 TP en EPU-E5-INC – Informatique C

- Année 2013-2014 : 78h TD
  - 6TD en EPU-C5-III – Introduction à l'informatique
  - 40TD en EPU-C6-IGV –Informatique générale-MTX3-AGRAL3-S2
  - 32TP en EPU-E5-INC – Informatique C EI2I-3 S

- Année 2014-2015 : 54h TD
  - 12CM et 36TP en EPU-I7-ICP – Langage C++

Paris le 23 septembre 2015

U.F.R. 919 – FACULTÉ D'INGÉNIERIE
Tour 55-65 – 2e étage porte 201A – case courrier 175
4, place Jussieu 75252 PARIS Cedex 05
Téléphone : 01 44 27 23 80 - Télécopie : 01 44 27 21 61

Paris, le 23 septembre 2015

Sylvain GUILLEY,
Professeur à Télécom ParisTech.

Objet : *Attestion de co-encadrement d'un stage M2.*

Ce document atteste que Pablo Rauzy a co-encadré pendant sa thèse de doctorat à TELECOM-ParisTech le stage de recherche de Martin Moreau, étudiant en Master 2 à l'UPMC, dans le cadre de la spécialité « *Sécurité Fiabilité et Performance du Numérique* ». Le stage portait sur la définition d'un modèle formel pour une contremesure dite d'« *extension de module* » permettant de réaliser des calculs sur des grands nombres avec une redondance, permettant ainsi la détection d'éventuelles erreurs.

Ce stage s'est très bien déroulé. A la fois le stagiaire a pu confirmer son goût pour la recherche (il va poursuivre son cursus par une thèse de doctorat auprès de l'IMDEA Software Institute à Madrid), et les résultats scientifiques obtenus sont novateurs. Par exemple, nous avons porté une protection (dite de Shamir) de *RSA* aux *courbes elliptiques*, qui nécessitent des calculs d'inversion modulaire. Cette étape peut conduire à des divisions par zéro si elle est réalisée de manière aveugle dans des anneaux tels que $\boldsymbol{Z}_{pr}$. Grâce au travail de M. Moreau, ce problème a été résolu en remplaçant l'inversion modulaire par une exponentiation. De plus, la protection appliquée à un calcul de multiplication scalaire sur courbe elliptique a été portée vers la bibliothèque mini-GMP, et programmée dans une carte d'évaluation ARM Cortex M4. Ce stage débouche sur la préparation d'un article à soumettre à la conférence PKC (Public Key Cryptography, workshop de l'IACR – International Association for Cryptology Research), et d'une partie d'un livre sur la protection des algorithmes cryptographiques à base de couplage.

Je confirme que Pablo a contribué de façon notable au succès de ce stage M2. Notamment, il a su transférer son savoir-faire en matière de méthodes formelles, puis guider Martin Moreau afin qu'il soit autonome et puisse apporter lui-même des contributions.

Sylvain Guilley.

INVENTEURS DU MONDE NUMÉRIQUE

A qui de droit

Recommandation pour la candidature de Monsieur Pablo Rauzy à la qualification aux fonctions de maître de conférences

Pablo Rauzy a commencé son postdoctorat au sein de l'équipe-projet Inria PRIVATICS en octobre dernier. Les recommandations que j'ai pu obtenir avant de lui attribuer ce poste étaient particulièrement élogieuses. Pablo Rauzy présente un profil assez unique, alliant capacité d'abstraction, aptitude à mettre ses idées en pratique et volonté de placer sa recherche dans un contexte social plus large. En attestent notamment sa liste de publications, ses nombreux développements ainsi que ses présentations et autres activités variées (notamment sur le thème de l'open access). J'ai pu moi-même vérifier toutes ces qualités au cours des trois premiers mois de post-doctorat de Pablo Rauzy qui ont permis de défricher un sujet complexe (notion de contrôle en matière de vie privée) et de poser les fondements d'une contribution sur le sujet.

Par ailleurs Pablo Rauzy a déjà beaucoup enseigné pendant sa thèse et ses qualités pédagogiques sont louées par ses collègues et étudiants.

Pour toutes ces raisons, je recommande fortement la candidature de Pablo Rauzy à la qualification aux fonctions de maître de conférences.

Villeurbanne, le 9 décembre 2015

Daniel Le Métayer
Directeur de Recherche Inria
Responsable de l'Inria Project Lab CAPPRIS

Inria Grenoble Rhône-Alpes
Antenne Lyon La Doua
Bâtiment CEI-2
56 boulevard Niels Bohr
69003 Villeurbanne

CENTRE DE RECHERCHE
GRENOBLE - RHÔNE-ALPES
Inovallée
655 avenue de l'Europe Montbonnot
38 334 Saint Ismier Cedex
Tél. : +33 (0)4 76 61 52 00
Fax : +33 (0)4 76 61 52 52
www.inria.fr

Service des Ressources Humaines
Affaire suivie par : Lila MEZIANI
Tél. : 04 76 61 52 80
Mail : lila.meziani@inria.fr

Réf. : RA/SRH/2015-557

CONTRAT DE TRAVAIL A DURÉE DÉTERMINÉE

Entre l'Institut national de recherche en informatique et en automatique représenté par son Président-directeur général, Monsieur Antoine PETIT, ci-après désigné « Inria » ;

Et Monsieur Pablo RAUZY, né le 30/07/1989 ci-après désigné par « le bénéficiaire » ;

Vu la loi n° 84-16 du 11 janvier 1984 modifiée portant dispositions statutaires relatives à la fonction publique de l'Etat et notamment son article 4.2 ;

Vu le décret n° 85-831 du 2 Août 1985 modifié portant organisation et fonctionnement de l'institut national de recherche en informatique et en automatique ;

Vu le décret n° 86-83 du 17 janvier 1986 modifié relatif aux dispositions générales applicables aux agents contractuels de l'Etat ;

Sous réserve de l'accord du Fonctionnaire sécurité défense ;

Il est convenu ce qui suit,

Article I – Objet :

Le bénéficiaire est recruté en qualité de post doctorant (emploi relevant de la catégorie « A »).

Le bénéficiaire est placé sous l'autorité hiérarchique du responsable de l'équipe projet PRIVATICS.
Il est affecté au Centre de recherche Grenoble - Rhône-Alpes et exerce ses fonctions au sein de l'équipe projet PRIVATICS.

Article II – Nature, durée et période d'essai :

Le présent contrat est à durée déterminée. Il prend effet à compter du 01/10/2015 et se terminera le 30/09/2016.

Il comprend une période d'essai d'un mois, éventuellement renouvelable, au cours de laquelle le présent contrat pourra être résilié par l'une ou l'autre des parties sans indemnité ni préavis.

Le présent contrat ne constitue pas un engagement à caractère permanent et ne confère en aucun cas au bénéficiaire le droit à une intégration dans le cadre des personnels statutaires d'Inria.

Il n'y aura pas lieu de verser une indemnité de fin de contrat.

Article III – Résidence administrative :

La résidence administrative du bénéficiaire se situe à Villeurbanne.

PR

A ce titre il pourra prétendre au remboursement partiel des frais de transport accordé aux agents, en fonction de leur résidence administrative.

Article IV – Rémunération et prise en charge financière :

Pour la durée du présent contrat, le bénéficiaire exercera ses fonctions à temps complet et percevra une rémunération mensuelle brute de 2621 euros.

Cette rémunération est indexée sur l'évolution de la valeur du point fonction publique. La valeur du point prise en référence est celle du 01/07/2010.

A cette rémunération s'ajoutera le cas échéant le supplément familial de traitement.

Article V – Couverture sociale :

Le bénéficiaire bénéficie des prestations du régime général de la sécurité sociale (assurance maladie, allocations familiales, accident du travail, maladie professionnelle, retraite).

Il bénéficie du régime de retraite complémentaire de l'IRCANTEC.

Il est assuré contre le risque de perte d'emploi, selon la réglementation de l'UNEDIC.

Il peut prétendre aux congés (maladie, maternité, paternité, adoption, parental, accident du travail, maladie professionnelle...) dans les conditions fixées par le décret du 17 janvier 1986 susvisé.

Article VI – Horaires, congés, déplacements :

En ce qui concerne les horaires de travail, la durée des congés annuels et les déplacements, le bénéficiaire du présent contrat est soumis aux règles applicables aux agents d'Inria. Les jours de congés annuels et RTT doivent être pris pendant la durée du contrat. Aucune indemnité ne sera due pour compenser les congés non utilisés du fait du bénéficiaire.

Le bénéficiaire est soumis à la réglementation sur les cumuls.

Article VII – Obligations de réserve et publications :

Le bénéficiaire est soumis aux obligations incombant à l'ensemble des agents d'Inria notamment à celle d'obéissance hiérarchique, d'obligation de réserve et de respect des règles en matière de sécurité informatique ; il est également tenu au secret professionnel à l'égard des tiers pour tous les faits, informations et documents dont il aura eu connaissance à l'occasion de l'exercice de ses fonctions au sein d'Inria. Cette dernière disposition demeure valable après son départ d'Inria.

Le bénéficiaire qui souhaite effectuer une publication doit solliciter de manière expresse, de l'autorité hiérarchique, l'autorisation de publier.

Durant son contrat, le bénéficiaire va avoir accès à des Informations Confidentielles, de manière orale ou écrite. Le bénéficiaire garantit qu'il ne divulguera pas ces Informations Confidentielles pendant la durée du contrat (article II) et pendant une période de cinq (5) ans à compter de son terme.

Sont considérées, au sens du présent article, comme une Information Confidentielle : toute information, et/ou toute donnée, de toute nature y compris juridique, financière et économique, transmise sous quelque forme qu'elle soit, incluant notamment tous documents écrits ou imprimés, tous échantillons,

PR

modèles, et/ou connaissances brevetables ou non, divulgués par Inria au bénéficiaire dans le cadre de l'exécution du présent contrat et sous réserve qu'Inria en ait indiqué de manière claire et non équivoque le caractère confidentiel ou dans le cas d'une divulgation orale qu'Inria ait fait connaître oralement le caractère confidentiel au moment de la divulgation et ait confirmé par écrit ce caractère dans un délai de trente (30) jours.

Le bénéficiaire s'engage à détruire les documents écrits en sa possession (en cas de communication d'Information Confidentielle sous forme écrite) dès la première demande d'Inria. Il enverra à Inria une attestation de destruction des documents dans un délai de quinze (15) jours suivant la destruction.

S'il s'avérait indispensable pour le bénéficiaire d'utiliser une Information Confidentielle, il en fera préalablement à toute utilisation, la demande à la direction du centre de recherche dont il relève.

Aucune divulgation d'Informations Confidentielles ne pourra être effectuée par le bénéficiaire sans avoir préalablement reçu une autorisation expresse de divulguer de la part d'Inria."

Article VIII – Propriété intellectuelle et industrielle :

Conformément aux dispositions de l'article L113-9 du Code de la propriété intellectuelle, les droits patrimoniaux sur les logiciels et la documentation y afférant, développés par le bénéficiaire au sein d'Inria dans le cadre de ses fonctions, sont la propriété d'Inria.

Conformément à l'article 131-3-1 du code de la propriété intellectuelle, les droits patrimoniaux sur les autres travaux protégés par le droit d'auteur, éventuellement créés par le bénéficiaire dans le cadre de ses fonctions, sont cédés de plein droit à Inria lorsque:

- les droits d'exploitation de ces autres travaux, à des fins non commerciales, sont nécessaires à l'accomplissement de la mission de service de public d'Inria, sous réserve de l'article 111-1 du code de la propriété intellectuelle.

Si Inria souhaite exploiter ces autres travaux à des fins commerciales, il disposera d'un droit de préférence, conformément à l'article 131-3-1 du code de propriété intellectuelle.

- ces autres travaux sont issus d'activités faisant l'objet d'un contrat avec une personne morale de droit privé. Dans cette hypothèse, la cession des droits d'auteurs sur ces autres travaux en faveur d'Inria est également valable pour une exploitation commerciale, conformément à l'article 131-3-1 du code de la propriété intellectuelle.

Conformément aux dispositions de l'article L611-7 du Code de la propriété intellectuelle, les droits de propriété industrielle sur les travaux réalisés par le bénéficiaire au sein d'Inria, dans le cadre de sa mission de recherche, sont automatiquement dévolus à Inria.
A ce titre, le bénéficiaire est tenu de déclarer à Inria l'existence de tous travaux de cette nature réalisés par lui, ainsi que toute information utile y afférant, y compris en vue d'un dépôt d'une demande de brevet portant sur lesdits travaux. Le bénéficiaire s'engage par ailleurs à s'abstenir de toute divulgation de nature à compromettre la validité de cette demande.

Il est précisé que les droits sur les bases de données élaborées au sein d'Inria, y compris celles sur lesquelles le bénéficiaire serait amené à contribuer, dans le cadre de ses fonctions, sont et restent la propriété d'Inria en tant que producteur de la base de données, conformément à l'article L341-1 du Code de la propriété intellectuelle.

Le bénéficiaire sera informé, au cours de son contrat et le cas échéant, à la fin de ce dernier, lorsque des travaux valorisables auxquels il aurait contribué auront vocation à faire l'objet d'un transfert de technologie, sous la forme d'une communication de savoir-faire. Dans cette hypothèse, le bénéficiaire sera tenu à une obligation de confidentialité et ne pourra pas, sauf accord expresse d'Inria, divulguer ledit savoir-faire jusqu'à ce que ce dernier soit tombé dans le domaine public.

PR

Article IX – Fin de contrat, démission, licenciement :

En matière de fin de contrat, démission, licenciement, le bénéficiaire est soumis aux dispositions du titre XI du décret du 17 janvier 1986 susvisé et notamment :

- à l'article 45 pour la fin du contrat ou son renouvellement,
- à l'article 48 pour la démission,
- aux articles 46, 47 et 49 pour le licenciement.

Le présent contrat pourra être résilié :

- sans préavis, à l'initiative de l'une ou l'autre des parties pendant la période d'essai suivant l'entrée en fonctions ou passé ce délai, en cas de faute grave, par décision unilatérale du président directeur général d'Inria.
- avec préavis :
  - à l'initiative du bénéficiaire du présent contrat,
  - à l'initiative du Président-directeur général d'Inria passé la période d'essai fixée par le présent article, pour des motifs réels et sérieux. En ce cas, le bénéficiaire sera informé des griefs portés contre lui et mis en mesure de présenter ses observations sur les faits qui lui sont reprochés.

Hormis le cas de faute grave, pour lequel le licenciement sans indemnités ni préavis peut être prononcé, la durée du préavis à respecter par l'une ou l'autre des parties est la suivante :

- huit jours si le bénéficiaire a moins de 6 mois de service,
- un mois, s'il a au moins 6 mois de service et moins de 2 ans,
- deux mois s'il a au moins 2 ans de service.

Le bénéficiaire
(signature précédée de la mention "lu et approuvé")

lu et approuvé

Fait à Montbonnot, le 31/08/2015
Pour le Président-directeur général et par délégation,

Pour le Président et par délégation
La responsable du pôle
Gestion administrative des Ressources Humaines

Aurélia Mouton

Formally Proved Security of Assembly Code Against Power Analysis

A Case Study on Balanced Logic

Pablo Rauzy Sylvain Guilley
Zakaria Najm
Institut Mines-Télécom ; Télécom ParisTech ; CNRS LTCI
firstname.*lastname*@telecom-paristech.fr

Abstract

In his keynote speech at CHES 2004, Kocher advocated that side-channel attacks were an illustration that formal cryptography was not as secure as it was believed because some assumptions (*e.g.*, no auxiliary information is available during the computation) were not modeled. This failure is caused by formal methods' focus on models rather than implementations. In this paper we present formal methods and tools for designing protected code and proving its security against power analysis. These formal methods avoid the discrepancy between the model and the implementation by working on the latter rather than on a high-level model. Indeed, our methods allow us (a) to automatically insert a power balancing countermeasure directly at the assembly level, and to prove the correctness of the induced code transformation; and (b) to prove that the obtained code is balanced with regard to a reasonable leakage model. We also show how to characterize the hardware to use the resources which maximize the relevancy of the model. The tools implementing our methods are then demonstrated in a case study on an 8-bit AVR smartcard for which we generate a provably protected PRESENT implementation that reveals to be at least 250 times more resistant to CPA attacks.

Keywords. Dual-rail with Precharge Logic (DPL), formal proof, static analysis, symbolic execution, implementation, DPA, CPA, smartcard, PRESENT, block cipher, Hamming distance, OCaml.

1 Introduction

The need to trust code is a clear and proved fact, but the code itself needs to be proved before it can be trusted. In applications such as cryptography or real-time systems, formal methods are used to prove functional properties on the critical parts of the code. Specifically in cryptography, some non-functional properties are also important, but are not typically certified by formal proofs yet. One example of such a property is the resistance to side-channel attacks. Side-channel attacks are a real world threat to cryptosystems; they exploit auxiliary information gathered from implementations through physical channels such as power consumption, electromagnetic radiations, or time, in order to extract sensitive information (*e.g.*, secret keys). The amount of leaked information depends on the implementation and as such appears difficult to model. As a matter of fact, physical leakages are usually not modeled when it comes to prove the security properties of a cryptographic algorithm. By applying formal methods directly on implementations we can avoid the discrepancy between the

model and the implementation. Formally proving non-functional security properties then becomes a matter of modeling the leakage itself. In this chapter we make a first step towards formally trustable cryptosystems, including for non-functional properties, by showing that modeling leakage and applying formal methods to implementations is feasible.

Many existing countermeasures against side-channel attacks are implemented at the hardware level, especially for smartcards. However, software level countermeasures are also very important, not only in embedded systems where the hardware cannot always be modified or updated, but also in the purely software world. For example, Zhang *et al.* [ZJRR12] recently extracted private keys using side-channel attacks against a target virtual machine running on the same physical server as their virtual machine. Side channels in software can also be found each time there are some non-logic behaviors (in the sense that it does not appear in the equations / control-flow modeling the program) such as timing or power consumption (refer to [KJJ99]), but also some software-specific information such as packet size for instance (refer to [MO12]).

In many cases where the cryptographic code is executed on secure elements (smartcards, TPM, tokens, *etc.*) side-channel and fault analyses are the most natural attack paths. A combination of signal processing and statistical techniques on the data obtained by side-channel analysis allows to build key hypotheses distinguishers. The protection against those attacks is necessary to ensure that secrets do not leak, and most secure elements are thus evaluated against those attacks. Usual certifications are the common criteria (ISO/IEC 15408), the FIPS 140-2 (ISO/IEC 19790), or proprietary schemes (EMVCo, CAST, *etc.*).

Power analysis. It is a form of side-channel attack in which the attacker measures the power consumption of a cryptographic device. *Simple Power Analysis* (SPA) consists in directly interpreting the electrical activity of the cryptosystem. On unprotected implementations it can for instance reveal the path taken by the code at branches even when timing attacks [KJJ96] cannot. *Differential Power Analysis* (DPA) [KJJ99] is more advanced: the attacker can compute the intermediate values within cryptographic computations by statistically analyzing data collected from multiple cryptographic operations. It is powerful in the sense that it does not require a precise model of the leakage, and thus works blind, *i.e.*, even if the implementation is blackbox. As suggested in the original DPA paper by Kocher *et al.* [KJJ99], power consumption is often modeled by Hamming weight of values or Hamming distance of values' updates as those are very correlated with actual measures. Also, when the leakage is little noisy and the implementation is software, *Algebraic Side-Channel Attack* (ASCA) [RS09] are possible; they consist in modelling the leakage by a set of Boolean equations, where the key bits are the only unknown variables [CFGR12].

Thwarting side-channel analysis is a complicated task, since an unprotected implementation leaks at every step. Simple and powerful attacks manage to exploit any bias. In practice, there are two ways to protect cryptosystems: "palliative" versus "curative" countermeasures. Palliative countermeasures attempt to make the attack more difficult, however without a theoretical foundation. They include variable clock, operations shuffling, and dummy encryptions among others (see also [GM11]). The lack of theoretical foundation make these countermeasures hard to formalize and thus not suitable for a safe certification process. Curative countermeasures aim at providing a leak-free implementation based on a security rationale. The two defense strategies are (a) make the leakage as decorrelated from the manipulated data as possible (*masking* [MOP06, Chp. 9]), or (b) make the leakage constant, irrespective of the manipulated data (hiding or *balancing* [MOP06, Chp. 7]).

Masking. Masking mixes the computation with random numbers, to make the leakage (at least in average) independent of the sensitive data. Advantages of masking are (*a priori*) the independence with respect to the leakage behavior of the hardware, and the existence of provably secure masking schemes [RP10]. There are two main drawbacks to masking. First of all, there is the possibility of high-order attacks (that examine the variance or the joint leakage); when the noise is low, ASCAs can be carried out on one single trace [RSVC09], despite the presence of the masks, that are just seen as more unknown variables, in addition to the key. Second, masking demands a greedy requirement for randomness (that is very costly to generate). Another concern with masking is the overhead it incurs in the computation time. For instance, a provable masking of AES-128 is reported in [RP10] to be 43 (resp. 90) times slower than the non-masked implementation with a 1st (resp. 2nd) order masking scheme. Further, recent studies have shown that masking cannot be analyzed independently from the execution platform: for example *glitches* are transient leakages that are likely to depend on more than one sensitive data, hence being high-order [MS06]. Indeed, a glitch occurs when there is a race between two signals, *i.e.*, when it involves more than one sensitive variable. Additionally, the implementation must be carefully scrutinized to check for the absence of *demasking* caused by overwriting a masked sensitive variable with its mask.

Balancing. Balancing requires a close collaboration between the hardware and the software: two indistinguishable resources, from a side-channel point of view, shall exist and be used according to a dual-rail protocol. *Dual-rail with Precharge Logic* (DPL) consists in precharging both resources, so that they are in a common state, and then setting one of the resources. Which resource has been set is unknown to the attacker, because both leak in indistinguishable ways (by hypothesis). This property is used by the DPL protocol to ensure that computations can be carried out without exploitable leakage [TV06].

Contributions. Dual-rail with Precharge Logic (DPL) is a simple protocol that may look easy to implement correctly; however, in the current context of awareness about cyber-threats, it becomes evident that (independent) formal tools that are able to *generate* and *verify* a "trusted" implementation have a strong value.

- We describe a design method for developing balanced assembly code by making it obey the DPL protocol. This method consists in automatically inserting the countermeasure and formally proving that the induced code transformation is correct (*i.e.*, semantic preserving).
- We present a formal method (using symbolic execution) to statically prove the absence of power consumption leakage in assembly code provided that the hardware it runs on satisfies a finite and limited set of requirements corresponding to our leakage model.
- We show how to characterize the hardware to run the DPL protocol on resources which maximize the relevancy of the leakage model.
- We provide a tool called paioli[1] which implements the automatic insertion of the DPL countermeasure in assembly code, and, independently, is able to statically prove the power balancing of a given assembly code.
- Finally, we demonstrate our methods and tool in a case study on a software implementation of the PRESENT [BKL+07] cipher running on an 8-bit AVR micro-controller. Our practical results are very encouraging: the provably balanced DPL protected implementation is *at*

[1] `http://pablo.rauzy.name/sensi/paioli.html`

least 250 times more resistant to power analysis attacks than the unprotected version while being only 3 times slower. The Signal-to-Noise Ratio (SNR) of the leakage is divided by approximately 16.

Related work. The use of formal methods is not widespread in the domain of implementations security. In cases where they exist, security proofs are usually done on mathematical models rather than implementations. An emblematic example is the Common Criteria [Con13], that bases its "formal" assurance evaluation levels on "Security Policy Model(s)" (class SPM) and not on implementation-level proofs. This means that it is the role of the implementers to ensure that their implementations fit the model, which is usually done by hand and is thus error-prone. For instance, some masking implementations have been proved; automatic tools for the insertion of masked code have even been prototyped [MOPT12]. However, masking relies a lot on randomness, which is a rare resource and is hard to formally capture. Thus, many aspects of the security are actually displaced in the randomness requirement rather that soundly proved. Moreover, in the field of masking, most proofs are still literate (*i.e.*, verified manually, not by a computer program). This has led to a recent security breach in what was supposed to be a proved [RP10] masking implementation [CGP+12]. Previous similar examples exist, *e.g.*, the purported high-order masking scheme [SP06], defeated one year after in [CPR07].

Timing and cache attacks are an exception as they benefit from the work of Köpf *et al.* [KB07, KD09]. Their tool, CacheAudit [DFK+13], implements formal methods that directly work on x86 binaries.

Since we started our work on DPL, others have worked on similar approaches. Independently, it has been shown that SNR reduction is possible with other encodings that are less costly, such as "dual-nibble" (Chen *et al.* [CESY14]) or "m-out-of-n" (Servant *et al.* [SDMB14]). However, it becomes admittedly much more difficult to balance the resources aimed at hiding one each other. Thus, there is a trade-off between performance (in terms of execution speed and code size) and security. In this chapter we propose a proof-of-concept of maximal security.

In this light it is easy to conclude that the use of formal methods to prove the security of implementations against power analysis is a need, and a technological enabler: it would guarantee that the instantiations of security principles are as strong as the security principles themselves.

Organization of the chapter. The DPL countermeasure is studied in Sec. 2. Sec. 3 details our method to balance assembly code and prove that the proposed transformation is correct. Sec. 4 explains the formal methods used to compute a proof of the absence of power consumption leakage. Sec. 5 is a practical case study using the PRESENT algorithm on an AVR micro-controller. Conclusions and perspectives are drawn in Sec. 6.

2 Dual-rail with Precharge Logic

Balancing (or hiding) countermeasures have been employed against side channels since early 2004, with dual-rail with precharge logic. The DPL countermeasure consists in computing on a redundant representation: each bit y is implemented as a pair $(y_{\text{False}}, y_{\text{True}})$. The bit pair is then used in a protocol made up of two phases:

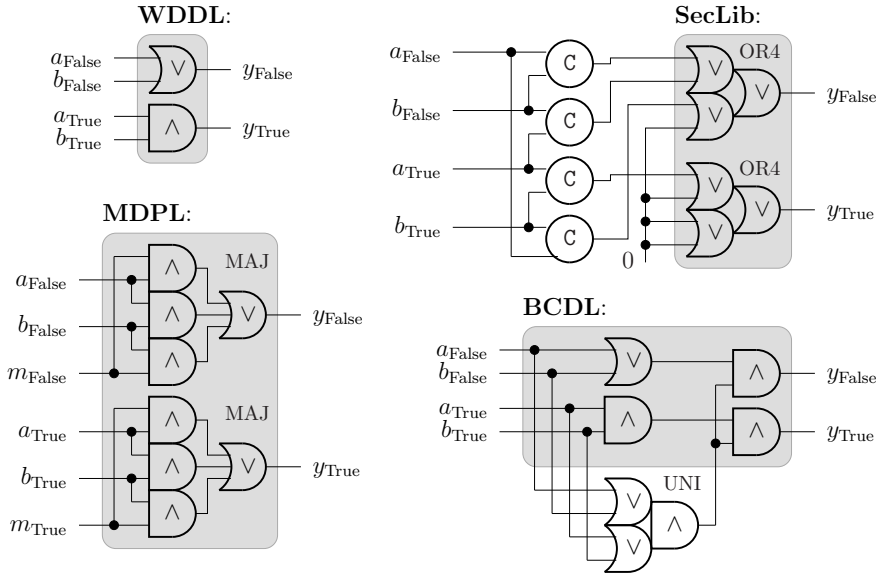


Figure 1: Four dual-rail with precharge logic styles.

1. a *precharge* phase, during which all the bit pairs are zeroized $(y_{\text{False}}, y_{\text{True}}) = (0, 0)$, such that the computation starts from a known reference state;
2. an *evaluation* phase, during which the $(y_{\text{False}}, y_{\text{True}})$ pair is equal to $(1, 0)$ if it carries the logical value 0, or $(0, 1)$ if it carries the logical value 1.

The value $(y_{\text{False}}, y_{\text{True}}) = (1, 1)$ is unused. As suggested in [MAM+03], it can serve as a *canary* to detect a fault. Besides, if a fault turns a $(1, 0)$ or $(0, 1)$ value into either $(0, 0)$ or $(1, 1)$, then the previous functional value has been forgiven. It is a type of infection, already mentioned in [IPSW06, SBG+09]. Unlike other infective countermeasure, DPL is not scary [BG13], in that it consists in an erasure. Indeed, the mutual information between the erased and the initial data is zero (provided only one bit out of a dual pair is modified).

2.1 State of the Art

Various DPL styles for electronic circuits have been proposed. Some of them, implementing the same logical *and* functionality, are represented in Fig. 1; many more variants exist, but these four are enough to illustrate our point. The reason for the multiplicity of styles is that the indistinguishability hypothesis on the two resources holding y_{False} and y_{True} values happens to be violated for various reasons, which leads to the development of dedicated hardware. A first asymmetry comes from the gates driving y_{False} and y_{True}. In *Wave Dynamic Differential Logic* (WDDL) [TV04a], these two gates are different: logical *or* versus logical *and*. Other logic styles are balanced with this respect. Then, the load of the gate shall also be similar. This can be achieved by careful place-and-route constraints [TV04b, GHMP05], that take care of having lines of the same length, and that furthermore do not interfere one with the other (phenomenon called "crosstalk"). As those are complex to implement exactly for all secure gates, the *Masked Dual-rail with Precharge Logic* (MDPL) [PM05] style has been proposed: instead of balancing exactly the lines carrying y_{False} and y_{True}, those are randomly swapped, according to a random bit, represented as a pair $(m_{\text{False}}, m_{\text{True}})$ to avoid it from leaking. Therefore, in this case, not only the computing gates are

the same (*viz.* a majority), but the routing is balanced thanks to the extra mask. However, it appeared that another asymmetry could be fatal to WDDL and MDPL: the gates pair could evaluate at different dates, depending on their input. It is important to mention that side-channel acquisitions are very accurate in timing (off-the-shelf oscilloscopes can sample at more than 1 Gsample/s, *i.e.*, at a higher rate than the clock period), but very inaccurate in space (*i.e.*, it is difficult to capture the leakage of an area smaller than about $1\,\mathrm{mm}^2$ without also recording the leakage from the surrounding logic). Therefore, two bits can hardly be measured separately. To avoid this issue, every gate has to include some synchronization logic. In Fig. 1, the "computation part" of the gates is represented in a grey box. The rest is synchronization logic. In SecLib [GCS+08], the synchronization can be achieved by Muller C-elements (represented with a symbol `C` [SEE98]), and act as a decoding of the inputs configuration. Another implementation, *Balanced Cell-based Differential Logic* (BCDL) [NBD+10], parallelize the synchronization with the computation.

2.2 DPL in Software

In this chapter, we want to run DPL on an off-the-shelf processor. Therefore, we must: (a) identify two similar resources that can hold true and false values in an indiscernible way for a side-channel attacker; (b) play the DPL protocol by ourselves, in software. We will deal with the former in Sec. 4.2. The rest of this section deals with the latter.

The difficulty of balancing the gates in hardware implementations is simplified in software. Indeed in software there are less resources than the thousands of gates that can be found in hardware (aimed at computing fast, with parallelism). Also, there is no such problem as early evaluation, since the processor executes one instruction after the other; therefore there are no unbalanced paths in timing. However, as noted by Hoogvorst *et al.* [HDD11], standard micro-processors cannot be used *as is* for our purpose: instructions may clobber the destination operand without precharge; arithmetic and logic instructions generate numbers of 1 and 0 which depend on the data.

To reproduce the DPL protocol in software requires (a) to work at the bit level, and (b) to duplicate (in positive and negative logic) the bit values. Every algorithm can be transformed so that all the manipulated values are bits (by the theorem of equivalence of universal Turing machines), so (*a*) is not a problem. Regarding (*b*), the idea is to use two bits in each register / memory cell to represent the logical value it holds. For instance using the two least significant bits, the logical value 1 could be encoded as 1 (`01`) and the logical value 0 as 2 (`10`). Then, any function on those bit values can be computed by a look-up table indexed by the concatenation of its operands. Each sensitive instruction can be replaced by a *DPL macro* which does the necessary precharge and fetch the result from the corresponding look-up table.

$$
\begin{array}{rcl}
r_1 & \leftarrow & r_0 \\
r_1 & \leftarrow & a \\
r_1 & \leftarrow & r_1 \wedge 3 \\
r_1 & \leftarrow & r_1 \ll 1 \\
r_1 & \leftarrow & r_1 \ll 1 \\
r_2 & \leftarrow & r_0 \\
r_2 & \leftarrow & b \\
r_2 & \leftarrow & r_2 \wedge 3 \\
r_1 & \leftarrow & r_1 \vee r_2 \\
r_3 & \leftarrow & r_0 \\
r_3 & \leftarrow & op[r_1] \\
d & \leftarrow & r_0 \\
d & \leftarrow & r_3
\end{array}
$$

Figure 2: DPL macro for $d = a$ op b.

Fig. 2 shows a DPL macro for the computation of $d = a$ op b, using the two least significant bits for the DPL encoding. The register r_0 is an always-zero register, a and b hold one DPL encoded bit, and *op* is the address in memory of the look-up table for the op operation.

This DPL macro assumes that before it starts the state of the program is a valid DPL state (*i.e.*, that a and b are of the form `/.+(01|10)/`[2]) and leaves it in a valid

[2] As a convenience, we use regular expressions notation.

DPL state to make the macros chainable.

The precharge instructions (like $r_1 \leftarrow r_0$) erase the content of their destination register or memory cell before use. If the erased datum is sensitive it is DPL encoded, thus the number of bit flips (*i.e.*, the Hamming distance of the update) is independent of the sensitive value. If the erased value is not sensitive (for example the round counter of a block cipher) then the number of bit flips is irrelevant. In both cases the power consumption provides no sensitive information.

The activity of the shift instructions (like $r_1 \leftarrow r_1 \ll 1$) is twice the number of DPL encoded bits in r_1 (and thus does not depend on the value when it is DPL encoded). The two most significant bits are shifted out and must be 0, *i.e.*, they cannot encode a DPL bit. The logical *or* instruction (as in $r_1 \leftarrow r_1 \vee r_2$) has a constant activity of one bit flip due to the alignment of its operands. The logical *and* instructions (like $r_1 \leftarrow r_1 \wedge 3$) flips as many bits as there are 1s after the two least significant bits (it's normally all zeros).

Accesses from/to the RAM (as in $r_3 \leftarrow op[r_1]$) cause as many bit flips as there are 1s in the transferred data, which is constant when DPL encoded. Of course, the position of the look-up table in the memory is also important. In order not to leak information during the addition of the offset ($op + r_1$ in our example), op must be a multiple of 16 so that its four least significant bits are 0 and the addition only flips the number of bits at 1 in r_1, which is constant since at this moment r_1 contains the concatenation of two DPL encoded bit values.

We could use other bits to store the DPL encoded value, for example the least and the third least significant bits. In this case a and b have to be of the form `/.+(0.1|1.0)/`, only one shift instruction would have been necessary, and the `and` instructions' mask would be 5 instead on 3.

3 Generation of DPL Protected Assembly Code

Here we present a generic method to protect assembly code against power analysis. To achieve that we implemented a tool (See App. A) which transforms assembly code to make it compliant with the DPL protocol described in Sec. 2.2. To be as universal as possible the tool works with a generic assembly language presented in Sec. 3.1. The details of the code transformation are given in Sec. 3.2. Finally, a proof of the correctness of this transformation is presented in Sec. 3.3.

We implemented paioli [1] using the OCaml[3] programming language, which type safety helps to prevent many bugs. On our PRESENT case-study, it runs in negligible time ($\ll 1$ second), both for DPL transformation and simulation, including balance verification. The unprotected (resp. DPL) bitslice AVR assembly file consists of 641 (resp. 1456) lines of code. We use nibble-wise jumps in each PRESENT operation, and an external loop over all rounds.

3.1 Generic Assembly Language

Our assembly language is generic in that it uses a restricted set of instructions that can be mapped to and from virtually any actual assembly language. It has the classical features of assembly languages: logical and arithmetical instructions, branching, labels, direct and indirect addressing. Fig. 3 gives the Backus–Naur Form (BNF) of the language while Fig. 4 gives the equivalent code of Fig. 2 as an example of its usage.

The semantics of the instructions are intuitive. For `Opcode2` and `Opcode3` the first operand is the destination and the other are the arguments. The `mov` instruction is used to copy registers,

[3] `http://ocaml.org/`

```
Prog    ::= ( Label? Inst? ( ';' <comment> )? '\n' )*
Label   ::= <label-name> ':'
Inst    ::= Opcode0
          | Branch1 Addr
          | Opcode2 Lval Val
          | Opcode3 Lval Val Val
          | Branch3 Val Val Addr
Opcode0 ::= 'nop'
Branch1 ::= 'jmp'
Opcode2 ::= 'not' | 'mov'
Opcode3 ::= 'and' | 'orr' | 'xor' | 'lsl' | 'lsr'
          | 'add' | 'mul'
Branch3 ::= 'beq' | 'bne'
Val     ::= Lval | '#' <immediate-value>
Lval    ::= 'r' <register-number>
          | '@' <memory-address>
          | '!' Val ( ',' <offset> )?
Addr    ::= '#' <absolute-code-address>
          | <label-name>
```

Figure 3: Generic assembly syntax (BNF).

load a value from memory, or store a value to memory depending on the form of its arguments. We remark that the instructions use the "`instr dest op1 op2`" format, which allows to map similar instructions from 32-bit processors directly, as well as instructions from 8-bit processors which only have two operands, by using the same register for `dest` and `op1` for instance.

3.2 Code Transformation

Bitsliced code. As seen in Sec. 2, DPL works at the bit level. Transforming code to make it DPL compliant thus requires this level of granularity. Bitslicing is possible on any algorithm[4], but we found that bitslicing an algorithm is hard to do automatically. In practice, every bitslice implementations we found were hand-crafted. However, since Biham presented his bitslice paper [Bih97], many block ciphers have been implemented in bitslice for performance reasons, which mitigate this concern. So, for the sake of simplicity, we assume that the input code is already bitsliced.

DPL macros expansion. This is the main point of the transformation of the code.

Definition 1 (Sensitive value)**.** A *value* is said *sensitive* if it depends on sensitive data. A sensitive data depends on the secret key or the plaintext[5].

[4]Intuitively, the proof invokes the Universal Turing Machines equivalence (those that work with only $\{0, 1\}$ as alphabet are as powerful as the others).

[5]Other works consider that a sensitive data must depend on both the secret key and the plaintext (as it is usually admitted in the "*only computation leaks*" paradigm; see for instance [RP10, §4.1]). Our definition is broader, in particular it also encompasses the random probing model [ISW03].

Table 1: Look-up tables for `and`, `or`, and `xor`.

idx	0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111
`and`	00 , 00 , 00 , 00 , 00 , 01 , 10 , 00 , 00 , 10 , 10 , 00 , 00 , 00 , 00 , 00
`or`	00 , 00 , 00 , 00 , 00 , 01 , 01 , 00 , 00 , 01 , 10 , 00 , 00 , 00 , 00 , 00
`xor`	00 , 00 , 00 , 00 , 00 , 10 , 01 , 00 , 00 , 01 , 10 , 00 , 00 , 00 , 00 , 00

Definition 2 (Sensitive instruction)**.** We say that an *instruction* is *sensitive* if it may modify the Hamming weight of a sensitive value.

All the sensitive instructions must be expanded to a DPL macro. Thus, all the sensitive data must be transformed too. Each literal ("immediate" values in assembly terms), memory cells that contain initialized constant data (look-up tables, etc.), and registers values need to be DPL encoded. For instance, using the two least significant bits, the 1s stay 1s (`01`) and the 0s become 2s (`10`).

Since the implementation is bitsliced, only the logical (bit level) operators are used in sensitive instructions (`and`, `or`, `xor`, `lsl`, `lsr`, and `not`). To respect the DPL protocol, `not` instructions are replaced by `xor` which inverse the positive logic and the negative logic bits of DPL encoded values. For instance if using the two least significant bits for the DPL encoding, `not` *a b* is replaced by `xor` *a b* `#3`. Bitsliced code never needs to use shift instructions since all bits are directly accessible.

Moreover, we currently run this code transformation only on block ciphers. Given that the code is supposed to be bitsliced, this means that the branching and arithmetic instructions are either not used or are used only in a deterministic way (*e.g.*, looping on the round counter) that does not depend on sensitive information.

```
mov r1 r0
mov r1 a
and r1 r1 #3
lsl r1 r1 #1
lsl r1 r1 #1
mov r2 r0
mov r2 b
and r2 r2 #3
orr r1 r1 r2
mov r3 r0
mov r3 !r1,op
mov d r0
mov d r3
```

Figure 4: DPL macro of Fig. 2 in assembly.

Thus, only `and`, `or`, and `xor` instructions need to be expanded to DPL macros such as the one shown in Fig. 4. This macro has the advantage that it actually uses two operands instructions only (when there are three operands in our generic assembly language, the destination is the same as one of the two others), which makes its instructions mappable one-to-one even with 8-bit assembly languages.

Look-up tables. As they appear in the DPL macro, the addresses of look-up tables are sensitive too. As seen in Sec. 2.2, the look-up tables must be located at an address which is a multiple of 16 so that the last four bits are available when adding the offset (in the case where we use the last four bits to place the two DPL encoded operands). Tab. 1 present the 16 values present in the look-up tables for `and`, `or`, and `xor`.

Values in the look-up tables which are not at DPL valid addresses, *i.e.*, addresses which are not a concatenation of `01` or `10` with `01` or `10`, are preferentially DPL invalid, *i.e.*, `00` or `11`. Like this if an error occurs during the execution (such as a fault injection for instance) it poisons the result and all the subsequent computations will be faulted too (infective computation).

Table 2: `and d a b`.

	a, b, d			
Before	0, 0, ?	0, 1, ?	1, 0, ?	1, 1, ?
After	0, 0, 0	0, 1, 0	1, 0, 0	1, 1, 1

Table 3: DPL `and d a b`.

a, b, d			
10, 10, ?	10, 01, ?	01, 10, ?	01, 01, ?
10, 10, 10	10, 01, 10	01, 10, 10	01, 01, 01

3.3 Correctness Proof of the Transformation

Formally proving the correctness of the transformation requires to define what we intend by "correct". Intuitively, it means that the transformed code does the "same thing" as the original one.

Definition 3 (Correct DPL transformation)**.** Let S be a valid state of the system (values in registers and memory). Let c be a sequence of instructions of the system. Let $\widehat{S}$ be the state of the system after the execution of c with state S, we denote that by $S \xrightarrow{c} \widehat{S}$. We write $dpl(S)$ for the DPL state (with DPL encoded values of the 1s and 0s in memory and registers) equivalent to the state S.
We say that c' is a *correct DPL transformation* of c if $S \xrightarrow{c} \widehat{S} \implies dpl(S) \xrightarrow{c'} dpl(\widehat{S})$.

Proposition 1 (Correctness of our code transformation)**.** *The expansion of the sensitive instructions into DPL macros such as presented in Sec. 2.2 is a correct DPL transformation.*

Proof. Let a and b be instructions. Let c be the code $a; b$ (instruction a followed by instruction b). Let X, Y, and Z be states of the program. If we have $X \xrightarrow{a} Y$ and $Y \xrightarrow{b} Z$, then we know that $X \xrightarrow{c} Z$ (by *transitivity*).

Let a' and b' be the DPL macro expansions of instructions a and b. Let c' be the DPL transformation of code c. Since the expansion into macros is done separately for each sensitive instruction, without any other dependencies, we know that c' is $a'; b'$.
If we have $dpl(X) \xrightarrow{a'} dpl(Y)$ and $dpl(Y) \xrightarrow{b'} dpl(Z)$, then we know that $dpl(X) \xrightarrow{c'} dpl(Z)$.

This means that a chain of correct transformations is a correct transformation. Thus, we only have to show that the DPL macro expansion is a correct transformation.

Let us start with the `and` operation. Since the code is bitsliced, there are only four possibilities. Tab. 2 shows these possibilities for the `and d a b` instruction.

Tab. 4 shows the evolution of the values of a, b, and d during the execution of the macro which `and d a b` expands to. We assume the look-up table for `and` is located at address *and*. Tab. 3 sums up the Tab. 4 in the same format as Tab. 2.

This proves that the DPL transformation of the `and` instructions are correct. The demonstration is similar for `or` and `xor` operations. □

The automatic DPL transformation of arbitrary assembly code has been implemented in our tool described in App. A.

4 Formally Proving the Absence of Leakage

Now that we know the DPL transformation is correct, we need to prove its efficiency security-wise. We prove the absence of leakage on the software, while obviously the leakage heavily depends on the hardware. Our proof thus makes an hypothesis on the hardware: we suppose that the bits

Table 4: Execution of the DPL macro expanded from `and d a b`.

a, b	10, 10				10, 01				01, 10				01, 01			
	d	r1	r2	r3	d	r1	r2	r3	d	r1	r2	r3	d	r1	r2	r3
`mov r1 r0`	?	0	?	?	?	0	?	?	?	0	?	?	?	0	?	?
`mov r1` a	?	10	?	?	?	10	?	?	?	01	?	?	?	01	?	?
`and r1 r1 #3`	?	10	?	?	?	10	?	?	?	01	?	?	?	01	?	?
`shl r1 r1 #1`	?	100	?	?	?	100	?	?	?	010	?	?	?	010	?	?
`shl r1 r1 #1`	?	1000	?	?	?	1000	?	?	?	0100	?	?	?	0100	?	?
`mov r2 r0`	?	1000	0	?	?	1000	0	?	?	0100	0	?	?	0100	0	?
`mov r2` b	?	1000	10	?	?	1000	01	?	?	0100	10	?	?	0100	01	?
`and r2 r2 #3`	?	1000	10	?	?	1000	01	?	?	0100	10	?	?	0100	01	?
`orr r1 r1 r2`	?	1010	10	?	?	1001	01	?	?	0110	10	?	?	0101	01	?
`mov r3 r0`	?	1010	10	0	?	1001	01	0	?	0110	10	0	?	0101	01	0
`mov r3 !r1,`and[6]	?	1010	10	10	?	1001	01	10	?	0110	10	10	?	0101	01	01
`mov` d `r0`	0	1010	10	10	0	1001	01	10	0	0110	10	10	0	0101	01	01
`mov` d `r3`	10	1010	10	10	10	1001	01	10	10	0110	10	10	01	0101	01	01

we use for the positive and negative logic in the DPL protocol leak the same amount. This may seem like an unreasonable hypothesis, since it is not true in general. However, the protection can be implemented in a soft CPU core (LatticeMicro32, OpenRISC, LEON2, etc.), that would be laid out in a Field-Programmable Gate Array (FPGA) or in an Application-Specific Integrated Circuit (ASIC) with special balancing constraints at place-and-route. The methodology follows the guidelines given by Chen *et al.* in [CSS13]. Moreover, we will show in Sec. 4.2 how it is possible, using stochastic profiling, to find bits which leakages are similar enough for the DPL countermeasure to be sufficiently efficient even on non-specialized hardware. That said, it is important to note that the difference in leakage between two bits of the same register should not be large enough for the attacker to break the DPL protection using SPA or ASCA.

Formally proving the balance of DPL code requires to properly define the notions we are using.

Definition 4 (Leakage model)**.** The attacker is able to measure the power consumption of parts of the cryptosystem. We model power consumption by the Hamming distance of values updates, *i.e.*, the number of bit flips. It is a commonly accepted model for power analysis, for instance with DPA [KJJ99] or Correlation Power Analysis (CPA) [BCO04]. We write $H(a, b)$ the Hamming distance between the values a and b.

Definition 5 (Constant activity)**.** The activity of a cryptosystem is said to be constant if its power consumption does not depend on the sensitive data and is thus always the same.
Formally, let $P(s)$ be a program which has s as parameter (*e.g.*, the key and the plaintext). According to our leakage model, a program $P(s)$ is of *constant activity* if:

- for every values s_1 and s_2 of the parameter s, for each cycle i, for every sensitive value v, v is updated at cycle i in the run of $P(s_1)$ if and only if it is updated also at cycle i in the run of $P(s_2)$;
- whenever an instruction modifies a sensitive value from v to v', then the value of $H(v, v')$ does not depend on s.

Remark 1. The first condition of Def. 5 mostly concerns leakage in the horizontal / time dimension, while the second condition mostly concerns leakage in the vertical / amplitude dimension.

Remark 2. The first condition of Def. 5 implies that the runs of the program $P(s)$ are constant in time for every s. This implies that a program of constant activity is not vulnerable to timing attacks, which is not so surprising given the similarity between SPA and timing attacks.

4.1 Computed Proof of Constant Activity

To statically determine if the code is correctly balanced (*i.e.*, that the activity of a given program is constant according to Def. 5), our tool relies on symbolic execution. The idea is to run the code of the program independently of the sensitive data. This is achieved by computing on sets of all the possible values instead of values directly. The symbolic execution terminates in our case because we are using the DPL protection on block ciphers, and we avoid combinatorial explosion thanks to bitslicing, as a value can initially be only 1 or 0 (or rather their DPL encoded counterparts). Indeed, bitsliced code only use logical instructions as explained in Sec. 3.2, which will always return a result in $\{0, 1\}$ when given two values in $\{0, 1\}$ as arguments.

Our tool implements an interpreter for our generic assembly language which work with sets of values. The interpreter is equipped to measure all the possible Hamming distances of each value

[6] See Tab. 1.

update, and all the possible Hamming weight of values. It watches updates in registers, in memory, and also in address buses (since the addresses may leak information when reading in look-up tables). If for one of these value updates there are different possible Hamming distances or Hamming weight, then we consider that there is a leak of information: the power consumption activity is not constant according to Def. 5.

Example. Let a be a register which can initially be either 0 or 1. Let b be a register which can initially be only 1. The execution of the instruction `orr` a a b will set the value of a to be all the possible results of $a \vee b$. In this example, the new set of possible values of a will be the singleton $\{1\}$ (since $0 \vee 1$ is 1 and $1 \vee 1$ is 1 too). The execution of this instruction only modified one value, that of a. However, the Hamming distance between the previous value of a and its new value can be either 0 (in case a was originally 1) or 1 (in case a was originally 0). Thus, we consider that there is a leak.

By running our interpreter on assembly code, we can statically determine if there are leakages or if the code is perfectly balanced. For instance for a block cipher, we initially set the key and the plaintext (*i.e.*, the sensitive data) to have all their possible values: all the memory cells containing the bits of the key and of the plaintext have the value $\{0, 1\}$ (which denotes the set of two elements: 0 and 1). Then the interpreter runs the code and outputs all possible leakage; if none are present, it means that the code is well balanced. Otherwise we know which instructions caused the leak, which is helpful for debugging, and also to locate sensitive portions of the code.

For an example in which the code is balanced, we can refer to the execution of the `and` DPL macro shown in Tab. 4. There we can see that the Hamming distance of the updates does not depend on the values of a and b. We also note that at the end of the execution (and actually, all along the execution) the Hamming weight of each value does not depend on a and b either. This allows to chain macros safely: each value is precharged with 0 before being written to.

4.2 Hardware Characterization

The DPL countermeasure relies on the fact that the pair of bits used to store the DPL encoded values leak the same way, *i.e.*, that their power consumptions are the same. This property is generally not true in non-specialized hardware. However, using the two closest bits (in terms of leakage) for the DPL protocol still helps reaching a better immunity to side-channel attacks, especially ASCAs that operate on a limited number of traces.

The idea is to compute the leakage level of each of the bits during the execution of the algorithm, in order to choose the two closest ones as the pair to use for the DPL protocol and thus ensure an optimal balance of the leakage. This is facilitated by the fact that the algorithm is bitsliced. Indeed, it allows to run the whole computation using only a chosen bit while all the others stay zero. We will see in Sec. 5.1 how we characterized our smartcard in practice.

5 Case Study: PRESENT on an ATmega163 AVR Micro-Controller

5.1 Profiling the ATmega163

We want to limit the size of the look-up tables used by the DPL macros. Thus, DPL macros need to be able to store two DPL encoded bits in the four consecutive bits of a register. This lets 13

possible DPL encoding layouts on 8-bit. Writing `X` for a bit that is used and `x` otherwise, we have:

1. `xxxxxxXX`,
2. `xxxxxXXx`,
3. `xxxxXXxx`,
4. `xxxXXxxx`,
5. `xxXXxxxx`,
6. `xXXxxxxx`,
7. `XXxxxxxx`,
8. `xxxxxXxX`,
9. `xxxxXxXx`,
10. `xxxXxXxx`,
11. `xxXxXxxx`,
12. `xXxXxxxx`,
13. `XxXxxxxx`.

As explained in Sec. 4.2, we want to use the pair of bits that have the closest leakage properties, and also which is the closest from the least significant bit, in order to limit the size of the look-up tables.

To profile the AVR chip (we are working with an *Atmel ATmega163 AVR* smartcard, which is *notoriously leaky*), we ran eight versions of an unprotected bitsliced implementation of PRESENT, each of them using only one of the 8 possible bits. We used the Normalized Inter-Class Variance (NICV) [BDGN14a], also called *coefficient of determination*, as a metric to evaluate the leakage level of the variables of each of the 8 versions. Let us denote by L the (noisy and non-injective) leakage associated with the manipulation of the sensitive value V, both seen as random variables; then the NICV is defined as the ratio between the inter-class and the total variance of the leakage, that is: $\text{NICV} = \frac{\mathsf{Var}[\mathbb{E}[L|V]]}{\mathsf{Var}[L]}$. By the Cauchy-Schwarz theorem, we have $0 \leqslant \text{NICV} \leqslant 1$; thus the NICV is an absolute leakage metric. A key advantage of NICV is that it detects leakage using public information like input plaintexts or output ciphertexts only. We used a fixed key and a variable plaintext on which applying NICV gave us the leakage level of all the intermediate variables in bijective relation with the plaintext (which are all the sensible data as seen in Def. 1). As we can see on the measures plotted in Fig. 5 (which can be found in App. B), the least significant bit leaks very differently from the others, which are roughly equivalent in terms of leakage[7]. Thus, we chose to use the `xxxxxXXx` DPL pattern to avoid the least significant bit (our goal here is not to use the optimal pair of bits but rather to demonstrate the added-value of the characterization).

5.2 Generating Balanced AVR Assembly

We wrote an AVR bitsliced implementation of PRESENT that uses the S-Box in 14 logic gates from Courtois *et al.* [CHM11]. This implementation was translated in our generic assembly language (see Sec. 3.1). The resulting code was balanced following the method discussed in Sec. 3, except that we used the DPL encoding layout adapted to our particular smartcard, as explained in Sec. 5.1. App. C presents the code of the adapted DPL macro. The balance of the DPL code was then verified as in Sec. 4. Finally, the verified code was mapped back to AVR assembly. All the code transformations and the verification were done automatically using our tool.

[7] These differences are due to the internal architecture of the chip, for which we don't have the specifications.

5.3 Cost of the Countermeasure

The table in Tab. 5 compares the performances of the DPL protected implementation of PRESENT with the original bitsliced version from which the protected one has been derived. The DPL countermeasure multiplies by 1.88 the size of the compiled code. This low factor can be explained by the numerous instructions which it is not necessary to transform (the whole permutation layer of the PRESENT algorithm is left as is for instance). The protected version uses 64 more bytes of memory (sparsely, for the DPL macro look-up tables). It is also only 3 times slower[8], or 24 times if we consider that the original bitsliced but unprotected code could operate on 8 blocks at a time.

Table 5: DPL cost.

	bitslice	DPL	cost
code (B)	1620	3056	$\times 1.88$
RAM (B)	288	352	$+64$
#cycles	78,403	235,427	$\times 3$

Note that these experimental results are only valid for the PRESENT algorithm on the *Atmel ATmega163 AVR* device we used. Further work is necessary to compare these results to those which would be obtained with other algorithms such as Advanced Encryption Standard (AES), and on other platforms such as ARM processors.

5.4 Attacks

We attacked three implementations of the PRESENT algorithm: a bitsliced but unprotected one, a DPL one using the two less significant bits, and a DPL one using two bits that are more balanced in term of leakage (as explained in Sec. 5.1). On each of these, we computed the success rate of using monobit CPA of the output of the S-Box as a model. The monobit model is relevant because only one bit of sensitive data is manipulated at each cycle since the algorithm is bitsliced, and also because each register is precharged at 0 before a new intermediate value is written to it, as per the DPL protocol prescribe. Note that this means we consider the resistance against first-order attacks only. Actually, we are precisely in the context of [MOS11], where the efficiency of correlation and Bayesian attacks gets close as soon as the number of queries required to perform a successful attack is large enough. This justifies our choice of the CPA for the attack evaluation.

The results are shown in Fig. 9 (which can be found in App. D.2). They demonstrate that the first DPL implementation is at least 10 times more resistant to first-order power analysis attacks (requiring almost 1,500 traces) than the unprotected one. The second DPL implementation, which takes the chip characterization into account, is 34 *times more resistant* (requiring more than 4,800 traces).

Interpreting these results requires to bear in mind that the *attacks setting was largely to the advantage of the attacker*. In fact, these results are very pessimistic: we used our knowledge of the key to select a narrow part of the traces where we knew that the attack would work, and we used the NICV [BDGN14a] to select the point where the SNR of the CPA attack is the highest (see similar use cases of NICV in [BDGN14b]). We did this so we could show the improvement in security due to the characterization of the hardware. Indeed, without this "cheating attacker" (for the lack of a better term), *i.e.*, when we use a monobit CPA taking into account the maximum

[8]Notice that PRESENT is inherently slow in software (optimized non-bitsliced assembly is reported to run in about 11,000 clock cycles on an *Atmel ATtiny 45* device [EGG+12]) because it is designed for hardware. Typically, the permutation layer is free in hardware, but requires many bit-level manipulations in software. Nonetheless, we emphasize that there are contexts where PRESENT must be supported, but no hardware accelerator is available.

of correlation over the full round, as a normal attacker would do, the unprotected implementation breaks using about 400 traces (resp. 138 for the "cheating attacker"), while the poorly balanced one is still not broken using $100,000$ traces (resp. about $1,500$). We do not have more traces than that so we can only say that with an experimental SNR of 15 (which is quite large so far), the security gain is more than $250\times$ and may be much higher with the hardware characterization taken into account as our results with the "cheating attacker" shows.

As a comparison[9], an unprotected AES on the same smartcard breaks in 15 traces, and in 336 traces with a first order masking scheme using less powerful attack setting (see success rates of masking in App. D.1), hence a security gain of $22\times$. Besides, we notice that our software DPL protection thwarts ASCAs. Indeed, ASCAs require a high signal to noise ratio on a single trace. This can happen both on unprotected and on masked implementation. However, our protection aims at theoretically cancelling the leakage, and practically manages to reduce it significantly, even when the chosen DPL bit pair is not optimal. Therefore, coupling software DPL with key-update [MSGR10] allows to both prevent against fast attacks on few traces (ASCAs) and against attacks that would require more traces (regular CPAs).

6 Conclusions and Perspectives

Contributions. We present a method to protect any bitsliced assembly code by transforming it to enforce the Dual-rail with Precharge Logic (DPL) protocol, which is a balancing countermeasure against power analysis. We provide a tool which automates this transformation. We also formally prove that this transformation is correct, *i.e.*, that it preserves the semantic of the program.

Independently, we show how to formally prove that assembly code is well balanced. Our tool is also able to use this technique to statically determine whether some arbitrary assembly code's power consumption activity is constant, *i.e.*, that it does not depend on the sensitive data. In this chapter we used the Hamming weight of values and the Hamming distance of values update as leakage models for power consumption, but our method is not tied to it and could work with any other leakage models that are computable. We present how to characterize the targeted hardware to make use of the resources which maximize the relevancy of our leakage model to run the DPL protocol.

We then applied our methods using our tool using an implementation of the PRESENT cipher on a real smartcard, which ensured that our methods and models are relevant in practice. In our case study, the provably balanced DPL protected implementation is at least 250 times more resistant to power analysis attacks than the unprotected version while being only 3 times slower. These figures could be better. Indeed, they do not take into account hardware characterization which helps the balancing a lot, as we were able to see with the "cheating attacker". Moreover, we have used the hardware characterization data grossly, only to show the added-value of the operation, which as expected is non-negligible. And of course interpreting our figures require to take into account that the *ATmega163*, the model of smartcard that we had at our disposal, is notoriously leaky.

These results show that software balancing countermeasures are realistic: our formally proved countermeasure is an order of magnitude less costly than the state of the art of formally proved masking [RP10].

[9] We insist that the comparison between two security gains is very platform-dependent. The figures we give are only valid on our specific setup. Of course, for different conditions, *e.g.*, lower signal-to-noise ratio, masking might become more secure than DPL.

Future work. The first and foremost future work surely is that our methods and tools need to be further tested in other experimental settings, across more hardware platforms, and using other cryptographic algorithms.

We did not try to optimize our PRESENT implementation (neither for speed nor space). However, automated proofs enable optimization: indeed, the security properties can be checked again after any optimization attempt (using proofs computation as non-regression tests, either for changes in the DPL transformation method, or for handcrafted optimizations of the generated DPL code).

Although the mapping from the internal assembly of our tool to the concrete assembly is straightforward, it would be better to have a formal correctness proof of the mapping.

Our work would also benefit from automated bitslicing, which would allow to automatically protect any assembly code with the DPL countermeasure. However, it is still a challenging issue.

Finally, the DPL countermeasure itself could be improved: the pair of bits used for the DPL protocol could change during the execution, or more simply it could be chosen at random for each execution in order to better balance the leakage among multiple traces. Besides, unused bits could be randomized instead of being zero in order to add noise on top of balancing, and thus reinforce the hypotheses we make on the hardware. An anonymous reviewer of the PROOFS 2014 workshop suggested that randomness could instead be used to mask the intermediate bits. Indeed, the reviewer thinks that switching bus lines may only increase noise, while masking variables may provide sound resistance, at least at first order. The resulting method would therefore: 1. gain both the 1st-order resistance of masking countermeasures and the significant flexibility of software-defined countermeasures; 2. still benefit from the increase of resistance resorting to the use of the DPL technique, as demonstrated by present chapter. This suggestion is of course only intuitive and lacks argumentation based on precise analysis and calculation.

We believe formal methods have a bright future concerning the certification of side-channel attacks countermeasures (including their implementation in assembly) for trustable cryptosystems.

References

[BCO04] Éric Brier, Christophe Clavier, and Francis Olivier. Correlation Power Analysis with a Leakage Model. In *CHES*, volume 3156 of *LNCS*, pages 16–29. Springer, August 11–13 2004. Cambridge, MA, USA.

[BDGN14a] Shivam Bhasin, Jean-Luc Danger, Sylvain Guilley, and Zakaria Najm. NICV: Normalized Inter-Class Variance for Detection of Side-Channel Leakage. In *International Symposium on Electromagnetic Compatibility (EMC '14 / Tokyo)*. IEEE, May 12-16 2014. Session OS09: EM Information Leakage. Hitotsubashi Hall (National Center of Sciences), Chiyoda, Tokyo, Japan.

[BDGN14b] Shivam Bhasin, Jean-Luc Danger, Sylvain Guilley, and Zakaria Najm. Side-channel Leakage and Trace Compression Using Normalized Inter-class Variance. In *Proceedings of the Third Workshop on Hardware and Architectural Support for Security and Privacy*, HASP '14, pages 7:1–7:9, New York, NY, USA, 2014. ACM.

[BG13] Alberto Battistello and Christophe Giraud. Fault Analysis of Infective AES Computations. In Wieland Fischer and Jörn-Marc Schmidt, editors, *2013 Workshop on Fault*

Diagnosis and Tolerance in Cryptography, Los Alamitos, CA, USA, August 20, 2013, pages 101–107. IEEE, 2013. Santa Barbara, CA, USA.

[Bih97] Eli Biham. A Fast New DES Implementation in Software. In Eli Biham, editor, *FSE*, volume 1267 of *Lecture Notes in Computer Science*, pages 260–272. Springer, 1997.

[BKL+07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and Charlotte Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In *CHES*, volume 4727 of *LNCS*, pages 450–466. Springer, September 10-13 2007. Vienna, Austria.

[CESY14] Cong Chen, Thomas Eisenbarth, Aria Shahverdi, and Xin Ye. Balanced Encoding to Mitigate Power Analysis: A Case Study. In *CARDIS*, Lecture Notes in Computer Science. Springer, November 2014. Paris, France.

[CFGR12] Claude Carlet, Jean-Charles Faugère, Christopher Goyet, and Guénaël Renault. Analysis of the algebraic side channel attack. *J. Cryptographic Engineering*, 2(1):45–62, 2012.

[CGP+12] Claude Carlet, Louis Goubin, Emmanuel Prouff, Michaël Quisquater, and Matthieu Rivain. Higher-Order Masking Schemes for S-Boxes. In Anne Canteaut, editor, *Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers*, volume 7549 of *Lecture Notes in Computer Science*, pages 366–384. Springer, 2012.

[CHM11] Nicolas Courtois, Daniel Hulme, and Theodosis Mourouzis. Solving Circuit Optimisation Problems in Cryptography and Cryptanalysis. *IACR Cryptology ePrint Archive*, 2011:475, 2011. (Also presented in SHARCS 2012, Washington DC, 17-18 March 2012, on page 179).

[Con13] Common Criteria Consortium. Common Criteria (*aka* CC) for Information Technology Security Evaluation (ISO/IEC 15408), 2013.
Website: `http://www.commoncriteriaportal.org/`.

[CPR07] Jean-Sébastien Coron, Emmanuel Prouff, and Matthieu Rivain. Side Channel Cryptanalysis of a Higher Order Masking Scheme. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES*, volume 4727 of *LNCS*, pages 28–44. Springer, 2007.

[CSS13] Zhimin Chen, Ambuj Sinha, and Patrick Schaumont. Using Virtual Secure Circuit to Protect Embedded Software from Side-Channel Attacks. *IEEE Trans. Computers*, 62(1):124–136, 2013.

[DFK+13] Goran Doychev, Dominik Feld, Boris Köpf, Laurent Mauborgne, and Jan Reineke. CacheAudit: A Tool for the Static Analysis of Cache Side Channels. *IACR Cryptology ePrint Archive*, 2013:253, 2013.

[EGG+12] Thomas Eisenbarth, Zheng Gong, Tim Güneysu, Stefan Heyse, Sebastiaan Indesteege, Stéphanie Kerckhof, François Koeune, Tomislav Nad, Thomas Plos, Francesco Regazzoni, François-Xavier Standaert, and Loïc van Oldeneel tot Oldenzeel. Compact Implementation and Performance Evaluation of Block Ciphers in ATtiny Devices. In

Aikaterini Mitrokotsa and Serge Vaudenay, editors, *AFRICACRYPT*, volume 7374 of *Lecture Notes in Computer Science*, pages 172–187. Springer, 2012.

[GCS+08] Sylvain Guilley, Sumanta Chaudhuri, Laurent Sauvage, Philippe Hoogvorst, Renaud Pacalet, and Guido Marco Bertoni. Security Evaluation of WDDL and SecLib Countermeasures against Power Attacks. *IEEE Transactions on Computers*, 57(11):1482–1497, nov 2008.

[GHMP05] Sylvain Guilley, Philippe Hoogvorst, Yves Mathieu, and Renaud Pacalet. The "Backend Duplication" Method. In *CHES*, volume 3659 of *LNCS*, pages 383–397. Springer, 2005. August 29th – September 1st, Edinburgh, Scotland, UK.

[GM11] Tim Güneysu and Amir Moradi. Generic side-channel countermeasures for reconfigurable devices. In Bart Preneel and Tsuyoshi Takagi, editors, *CHES*, volume 6917 of *LNCS*, pages 33–48. Springer, 2011.

[HDD11] Philippe Hoogvorst, Jean-Luc Danger, and Guillaume Duc. Software Implementation of Dual-Rail Representation. In *COSADE*, February 24-25 2011. Darmstadt, Germany.

[IPSW06] Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private Circuits II: Keeping Secrets in Tamperable Circuits. In *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 308–327. Springer, May 28 – June 1 2006. St. Petersburg, Russia.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private Circuits: Securing Hardware against Probing Attacks. In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, August 17–21 2003. Santa Barbara, California, USA.

[KB07] Boris Köpf and David A. Basin. An information-theoretic model for adaptive side-channel attacks. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM Conference on Computer and Communications Security*, pages 286–296. ACM, 2007.

[KD09] Boris Köpf and Markus Dürmuth. A provably secure and efficient countermeasure against timing attacks. In *CSF*, pages 324–335. IEEE Computer Society, 2009.

[KJJ96] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Proceedings of CRYPTO'96*, volume 1109 of *LNCS*, pages 104–113. Springer-Verlag, 1996.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *Proceedings of CRYPTO'99*, volume 1666 of *LNCS*, pages 388–397. Springer-Verlag, 1999.

[MAM+03] Simon Moore, Ross Anderson, Robert Mullins, George Taylor, and Jacques J.A. Fournier. Balanced Self-Checking Asynchronous Logic for Smart Card Applications. *Journal of Microprocessors and Microsystems*, 27(9):421–430, October 2003.

[MO12] Luke Mather and Elisabeth Oswald. Pinpointing side-channel information leaks in web applications. *J. Cryptographic Engineering*, 2(3):161–177, 2012.

[MOP06] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, December 2006. ISBN 0-387-30857-1, `http://www.dpabook.org/`.

[MOPT12] Andrew Moss, Elisabeth Oswald, Dan Page, and Michael Tunstall. Compiler Assisted Masking. In Emmanuel Prouff and Patrick Schaumont, editors, *CHES*, volume 7428 of *LNCS*, pages 58–75. Springer, 2012.

[MOS11] Stefan Mangard, Elisabeth Oswald, and François-Xavier Standaert. One for All - All for One: Unifying Standard DPA Attacks. *Information Security, IET*, 5(2):100–111, 2011. ISSN: 1751-8709 ; Digital Object Identifier: 10.1049/iet-ifs.2010.0096.

[MS06] Stefan Mangard and Kai Schramm. Pinpointing the Side-Channel Leakage of Masked AES Hardware Implementations. In *CHES*, volume 4249 of *LNCS*, pages 76–90. Springer, October 10-13 2006. Yokohama, Japan.

[MSGR10] Marcel Medwed, François-Xavier Standaert, Johann Großschädl, and Francesco Regazzoni. Fresh Re-Keying: Security against Side-Channel and Fault Attacks for Low-Cost Devices. In *AFRICACRYPT*, volume 6055 of *LNCS*, pages 279–296. Springer, May 03-06 2010. Stellenbosch, South Africa. DOI: 10.1007/978-3-642-12678-9_17.

[NBD+10] Maxime Nassar, Shivam Bhasin, Jean-Luc Danger, Guillaume Duc, and Sylvain Guilley. BCDL: A high performance balanced DPL with global precharge and without early-evaluation. In *DATE'10*, pages 849–854. IEEE Computer Society, March 8-12 2010. Dresden, Germany.

[PM05] Thomas Popp and Stefan Mangard. Masked Dual-Rail Pre-charge Logic: DPA-Resistance Without Routing Constraints. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, volume 3659 of *LNCS*, pages 172–186. Springer, 2005.

[RP10] Matthieu Rivain and Emmanuel Prouff. Provably Secure Higher-Order Masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *CHES*, volume 6225 of *LNCS*, pages 413–427. Springer, 2010.

[RS09] Mathieu Renauld and François-Xavier Standaert. Algebraic Side-Channel Attacks. In Feng Bao, Moti Yung, Dongdai Lin, and Jiwu Jing, editors, *Inscrypt*, volume 6151 of *Lecture Notes in Computer Science*, pages 393–410. Springer, 2009.

[RSVC09] Mathieu Renauld, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Algebraic Side-Channel Attacks on the AES: Why Time also Matters in DPA. In *CHES*, volume 5747 of *Lecture Notes in Computer Science*, pages 97–111. Springer, September 6-9 2009. Lausanne, Switzerland.

[SBG+09] Nidhal Selmane, Shivam Bhasin, Sylvain Guilley, Tarik Graba, and Jean-Luc Danger. WDDL is Protected Against Setup Time Violation Attacks. In *FDTC*, pages 73–83. IEEE Computer Society, September 6th 2009. In conjunction with CHES'09,

Lausanne, Switzerland. DOI: 10.1109/FDTC.2009.40; Online version: `http://hal.archives-ouvertes.fr/hal-00410135/en/`.

[SDMB14] Victor Servant, Nicolas Debande, Houssem Maghrebi, and Julien Bringer. Study of a Novel Software Constant Weight Implementation. In *CARDIS*, Lecture Notes in Computer Science. Springer, November 2014. Paris, France.

[SEE98] Maitham Shams, Jo. C. Ebergen, and Mohamed I. Elmasry. Modeling and comparing CMOS implementations of the C-Element. *IEEE Transactions on VLSI Systems*, 6(4):563–567, December 1998.

[SP06] Kai Schramm and Christof Paar. Higher Order Masking of the AES. In David Pointcheval, editor, *CT-RSA*, volume 3860 of *LNCS*, pages 208–225. Springer, 2006.

[TPR13] Adrian Thillard, Emmanuel Prouff, and Thomas Roche. Success through Confidence: Evaluating the Effectiveness of a Side-Channel Attack. In Guido Bertoni and Jean-Sébastien Coron, editors, *CHES*, volume 8086 of *Lecture Notes in Computer Science*, pages 21–36. Springer, 2013.

[TV04a] Kris Tiri and Ingrid Verbauwhede. A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In *DATE'04*, pages 246–251. IEEE Computer Society, February 2004. Paris, France. DOI: 10.1109/DATE.2004.1268856.

[TV04b] Kris Tiri and Ingrid Verbauwhede. Place and Route for Secure Standard Cell Design. In Kluwer, editor, *Proceedings of WCC / CARDIS*, pages 143–158, Aug 2004. Toulouse, France.

[TV06] Kris Tiri and Ingrid Verbauwhede. A digital design flow for secure integrated circuits. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 25(7):1197–1208, 2006.

[ZJRR12] Yinqian Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Cross-VM side channels and their use to extract private keys. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM Conference on Computer and Communications Security*, pages 305–316. ACM, 2012.

A paioli

The goal of paioli[10] (*Power Analysis Immunity by Offsetting Leakage Intensity*) is to protect assembly code against power analysis attacks such as DPA (differential power analysis) and CPA (correlation power analysis), and to formally prove the efficiency of the protection. To this end, it implements the automatic insertion of a balancing countermeasure, namely DPL (dual-rail with precharge logic), in assembly code (for now limited to bitsliced block-cipher type of algorithms). Independently, it is able to statically verify if the power consumption of a given assembly code is correctly balanced with regard to a leakage model (*e.g.*, the Hamming weight of values, or the Hamming distance of values updates).

```
paioli [options] <input-file>
    -bf Bit to use as F is DPL macros (default: 1)
    -bt Bit to use as T is DPL macros (default: 0)
    -po Less significant bit of the DPL pattern for DPL LUT access
        (default: 0)
    -cl Compact the DPL look-up table (LUT) if present
    -la Address in memory where to put the DPL LUT (default: 0)
    -r1 Register number of one of the three used by DPL macros
        (default: 20)
    -r2 Register number of one of the three used by DPL macros
        (default: 21)
    -r3 Register number of one of the three used by DPL macros
        (default: 22)
    -a  Adapter for custom assembly language
    -o  asm output (default: no output)
    -l  Only check syntax if present
    -d  Perform DPL transformation of the code if present
    -v  Perform leakage verification if present
    -s  Perform simulation if present
    -r  Register count for simulation (default: 32)
    -m  Memory size for simulation (default: 1024)
    -M  range of memory to display after simulation
    -R  range of registers to display after simulation
```

The rest of this section details its features.

Adapters. To easily adapt it to any assembly language, it has a system of plugins (which we call "adapters") that allows to easily write a parser and a pretty-printer for any language and to use them instead of the internal parser and pretty-printer (which are made for the internal language we use, see Sec. 3.1) without having to recompile the whole tool.

DPL transformation. If asked so, paioli is able to automatically apply the DPL transformation as explained in Sec. 3.2. It takes as arguments which bits to use for the DPL protocol, the offset at which to place the pattern for look-up tables (for example, we used an offset of 1 to avoid resorting to the least significant bit which leaks differently), and where in memory should the look-up tables start. Given these parameters, the tool verifies that they are valid and consistent according to the DPL protocol, and then it generates the DPL balanced code corresponding to the input code,

[10] `http://pablo.rauzy.name/sensi/paioli.html`

including the code for look-up tables initialization. Optionally, the tool is able to compact the look-up tables (since they are sparse), still making sure that their addresses respect the DPL protocol (Sec. 2.2).

Simulation. If asked so, paioli can simulate the execution of the code after its optional DPL transformation. The simulator is equipped to do the balance verification proof (see Sec. 4) but it is not mandatory to do the balance analysis when running it. It takes as parameters the size of the memory and the number of register to use, and initializes them to the set of two DPL encoded values of 1 and 0 corresponding to the given DPL parameters. The tool can optionally display the content of selected portions of the memory or of chosen registers after execution, which is useful for inspection and debugging purpose for example.

Balance verification. The formal verification of the balance of the code is an essential functionality of the tool. Indeed, bugs occur even when having a thorough and comprehensive specification, thus we believe that it is not sufficient to have a precise and formally proven method for generating protected code, but that the results should be independently verified (see Sec. 4).

B Characterization of the Atmel ATmega163 AVR Micro-Controller

Fig. 5 shows the leakage level computed using NICV [BDGN14a] for each bit of the *Atmel ATmega163 AVR* smartcard that we used for our tests (see Sec. 5.1). We can see the first bit leaks very differently from the others. Thus it is not a good candidate to appear in the bit pair used for the DPL protocol.

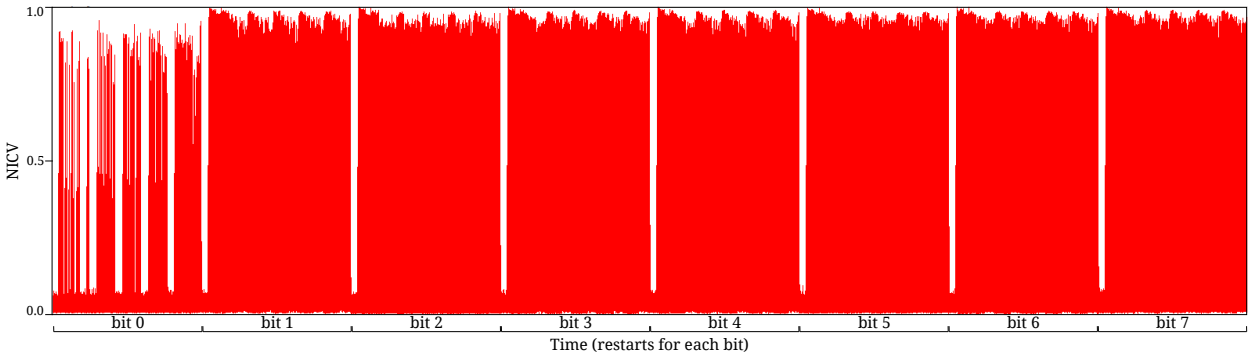
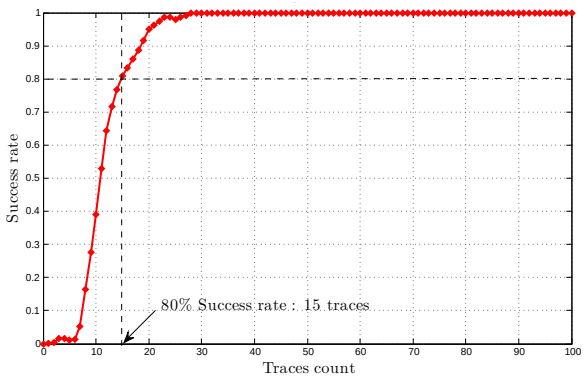


Figure 5: Leakage during unprotected encryption for each bit on *ATmega163*.

C DPL Macro for the AVR Micro-Controller

Once we profiled our smartcard as described in Sec. 5.1, we decided to use the bits 1 and 2 for the DPL protocol (`xxxxxXXx`), that is, the DPL value of 1 becomes 2 and the DPL value of 0 becomes 4. To avoid using the least significant bit (which leaks very differently from the others), we decided to align the two DPL bits for look-up table access starting on the bit 1 rather than 0 (`xxxXXXXx`). With these settings, the DPL macro automatically generated by paioli is presented in Fig. 6 (it

(a) Univariate CPA attack on unprotected AES.



(b) Bi-variate 2O-CPA on 1st-order protected AES.

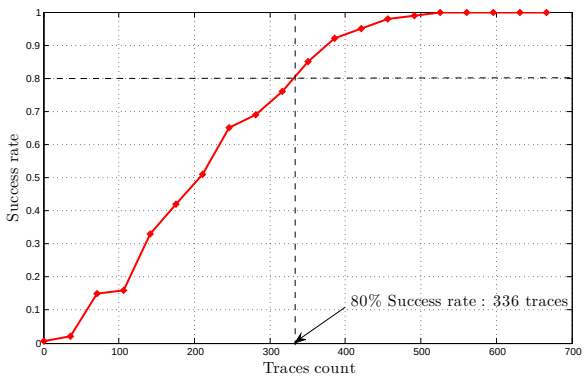


Figure 7: Attacking AES on the *ATmega163*: success rates.

follows the same conventions as Fig. 2). As we can see the only modification is the mask applied in the logical *and* instructions which is now 6 instead of 3 to reflect the new DPL pattern.

Note that the least significant bit is now unused by the DPL protocol and allowed paioli to compact the look-up tables used by the DPL macros. Indeed, their addresses need to be of the form `/.+0000./` leaving the least significant bit free and thus allowing to interleave two look-up tables one on another without overlapping of their actually used cells (see Sec. 3.2).

$$
\begin{array}{rcl}
r_1 & \leftarrow & r_0 \\
r_1 & \leftarrow & a \\
r_1 & \leftarrow & r_1 \wedge 6 \\
r_1 & \leftarrow & r_1 \ll 1 \\
r_1 & \leftarrow & r_1 \ll 1 \\
r_2 & \leftarrow & r_0 \\
r_2 & \leftarrow & b \\
r_2 & \leftarrow & r_2 \wedge 6 \\
r_1 & \leftarrow & r_1 \vee r_2 \\
r_3 & \leftarrow & r_0 \\
r_3 & \leftarrow & op[r_1] \\
d & \leftarrow & r_0 \\
d & \leftarrow & r_3
\end{array}
$$

Figure 6: DPL macro for $d = a$ op b on the *ATmega163*.

D Attacks

D.1 Attack results on masking (AES)

For the sake of comparison, we provide attack results on the same smartcard tested with the same setup. Figure 7 shows the success rate for the attack on the first byte of an AES.

We estimate the number of traces for a successful attack as the abscissa where the success rate curve first intersects the 80% horizontal line.

D.2 Attack results on DPL (PRESENT)

Fig. 9 shows the success rates and the correlation curves when attacking our three implementations of PRESENT. The sensitive variable we consider is in line with the choice of Kocher *et al.* in their CRYPTO'99 paper [KJJ99]: it is the least significant bit of the output of the substitution boxes (that are 4×4 in PRESENT).

In Fig. 8, we give, for the unprotected bitslice implementation, the correspondence between the operations of PRESENT and the NICV trace. The zones of largest NICV correspond to operations that access (read or write) sensitive data in RAM. To make the attacks more powerful, they are not done on the maximal correlation point over the full first round of PRESENT[11] (500,000 samples),

[11]Note that using the maximum correlation point to attack the DPL implementations resulted in the success

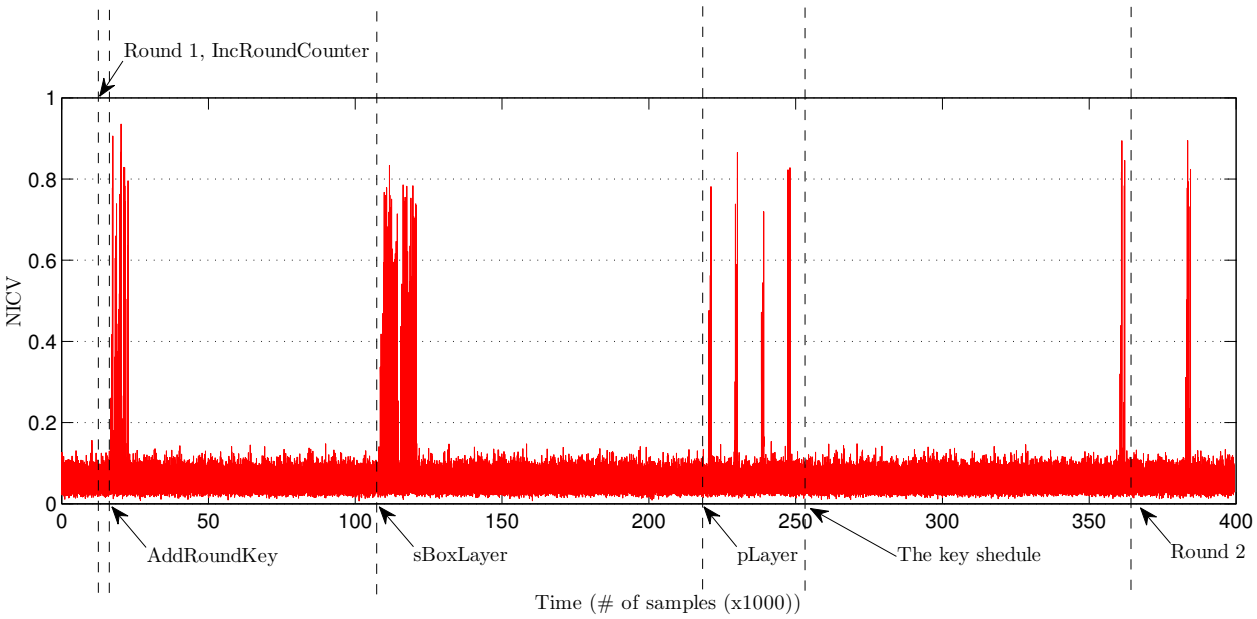


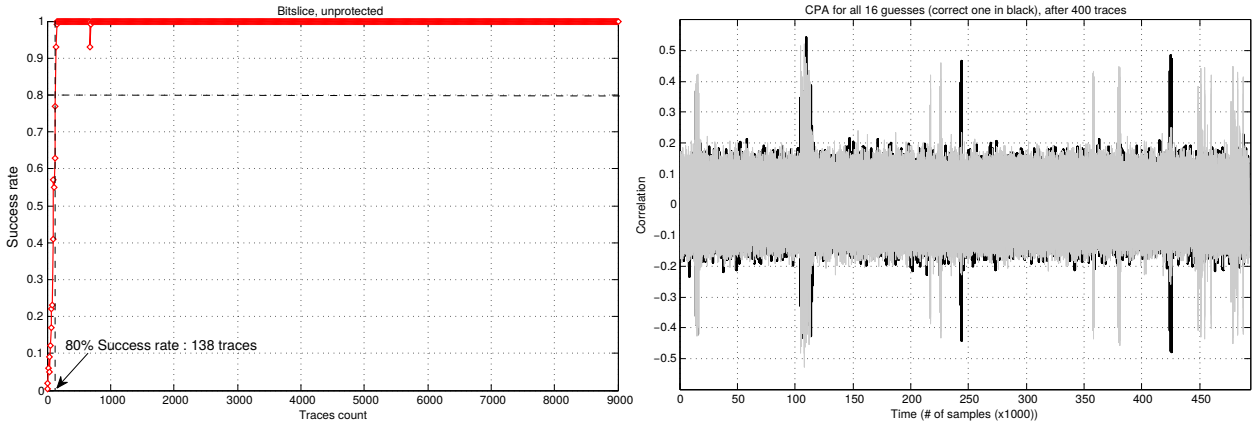
Figure 8: Correspondence between NICV and the instructions of PRESENT.

but rather on a smaller interval (of only 140 samples, *i.e.*, one clock period of the device) of high potential leakage revealed by the NICV computations, namely sBoxLayer.

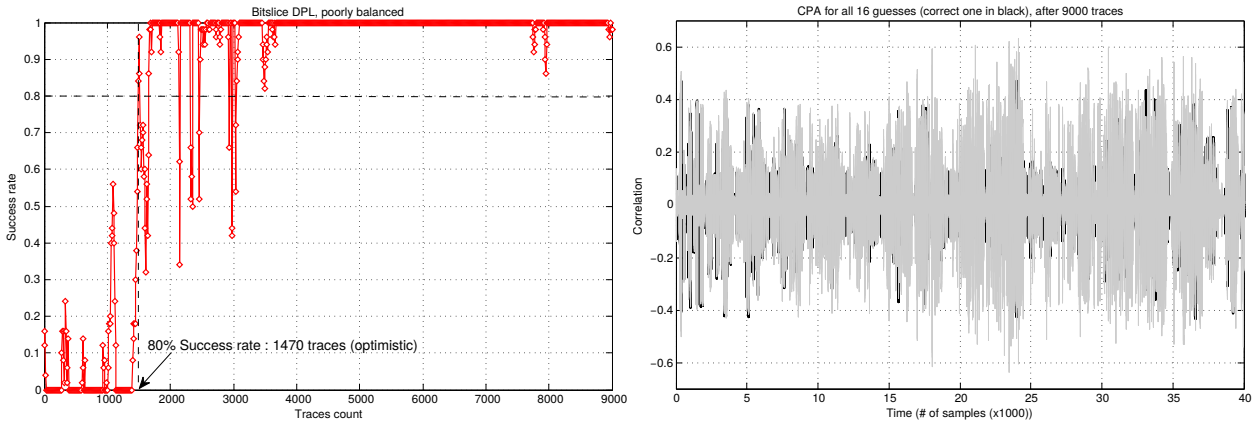
This makes the attack much more powerful and has to be taken into account when interpreting its results. In fact, the results we present are very pessimistic: we used our knowledge of the key to select a narrow part of the traces where we knew that the attack would work, and we used the NICV [BDGN14a] to select the point where the SNR of the CPA attack is the highest. We did this so we could show the improvement in security due to the characterization of the hardware. Indeed, without this "cheating attacker" (for the lack of a better term), *i.e.*, when we use a monobit CPA taking into account the maximum of correlation over the full round, as a normal attacker would do, the unprotected implementation breaks using about 400 traces (resp. 138 for the "cheating attacker"), while the poorly balanced one is still not broken using $100,000$ traces (resp. about $1,500$). We do not have more traces than that so we can only say that with an experimental SNR of 15 (which is quite large so far), the security gain is more than $250\times$ and may be much higher with the hardware characterization taken into account as our results with the "cheating attacker" shows. Another way of understanding the 250-fold data complexity increase for the CPA is to turn this figure into a reduction of the SNR: according to [TPR13, BDGN14b], our DPL countermeasure has attenuated the SNR by a factor of at least $\sqrt{250} \approx 16$.

rate remaining always at $\approx 1/16$ (there are 2^4 key guesses in PRESENT when targeting the first round, because the substitution boxes are 4×4) in average (at least on the number of traces we had $(100,000)$) on both on them.

(a) Monobit CPA attack on unprotected bitslice implementation.



(b) Monobit CPA attack on poorly balanced DPL implementation (bits 0 and 1).



(c) Monobit CPA attack on better balanced DPL implementation (bits 1 and 2).

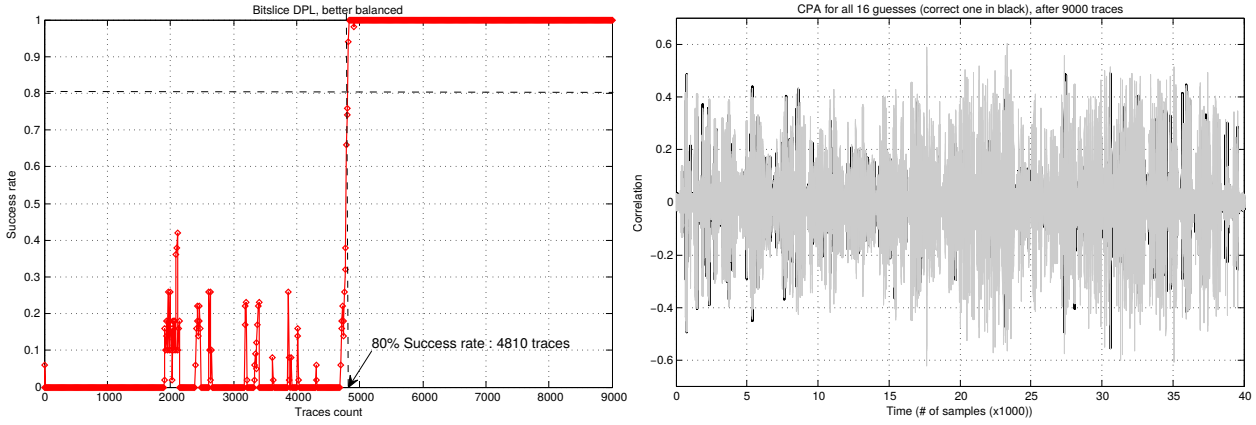


Figure 9: Attacks on our three implementations of PRESENT;
Left: success rates (estimated with 100 attacks/step), and
Right: CPA curves (whole first round in (a), and only sBoxLayer for (b) and (c)).

Countermeasures Against High-Order Fault-Injection Attacks on CRT-RSA

Pablo Rauzy and Sylvain Guilley
Institut Mines-Télécom ; Télécom ParisTech ; CNRS LTCI
{*firstname.lastname*}@telecom-paristech.fr

Abstract

In this paper we study the existing CRT-RSA countermeasures against fault-injection attacks. In an attempt to classify them we get to achieve deep understanding of how they work. We show that the many countermeasures that we study (and their variations) actually share a number of common features, but optimize them in different ways. We also show that there is no conceptual distinction between test-based and infective countermeasures and how either one can be transformed into the other. Furthermore, we show that faults on the code (skipping instructions) can be captured by considering only faults on the data. These intermediate results allow us to improve the state of the art in several ways: (a) we fix an existing and that was known to be broken countermeasure (namely the one from Shamir); (b) we drastically optimize an existing countermeasure (namely the one from Vigilant) which we reduce to 3 tests instead of 9 in its original version, and prove that it resists not only one fault but also an arbitrary number of randomizing faults; (c) we also show how to upgrade countermeasures to resist any given number of faults: given a correct first-order countermeasure, we present a way to design a provable high-order countermeasure (for a well-defined and reasonable fault model). Finally, we pave the way for a generic approach against fault attacks for any modular arithmetic computations, and thus for the automatic insertion of countermeasures.

1 Introduction

Private information protection is a highly demanded feature, especially in the current context of global defiance against most infrastructures, assumed to be controlled by governmental agencies. Properly used cryptography is known to be a key building block for secure information exchange. However, in addition to the threat of cyber-attacks, implementation-level hacks must also be considered seriously. This article deals specifically with the protection of a *decryption* or *signature* crypto-system (called RSA [RSA78]) in the presence of hardware attacks (*e.g.*, we assume the attacker can alter the RSA computation while it is being executed).

It is known since 1997 (with the BellCoRe attack by Boneh *et al.* [BDL97]) that injecting faults during the computation of CRT-RSA (CRT for "Chinese Remainder Theorem") could yield to malformed signatures that expose the prime factors (p and q) of the public modulus ($N = p \cdot q$). Notwithstanding, computing without the fourfold acceleration conveyed by the CRT optimization is definitely not an option in practical applications. Therefore, many countermeasures have appeared. Most of the existing countermeasures were designed with an attack-model consisting in a single fault injection. The remaining few attempts to protect against second-order fault attacks (*i.e.*, attacks with two faults).

Looking at the history of the development of countermeasures against the BellCoRe attack, we see that many countermeasures are actually broken in the first place. Some of them were fixed by their authors and/or other people, such as the countermeasure proposed by Vigilant [Vig08a], which was fixed by Coron *et al.* [CGM+10] and then simplified by Rauzy & Guilley [RG14b]; some simply abandoned, such as the one by Shamir [Sha99]. Second-order countermeasures are no exception to that rule, as demonstrated with the countermeasure proposed by Ciet & Joye [CJ05], which was fixed later by Dottax *et al.* [DGRS09]. Such mistakes can be explained by two main points:

- the almost nonexistent use of formal methods in the field of implementation security, which can itself be explained by the difficulty to properly model the physical properties of an implementation which are necessary to study side-channel leakages and fault-injection effects;
- the fact that most countermeasures were developed by trial-and-error engineering, accumulating layers of intermediate computations and verifications to patch weaknesses until a fixed point was reached, even if the inner workings of the countermeasure were not fully understood.

Given their development process, it is likely the case that existing second-order countermeasures would not resist third-order attacks, and strengthening them against such attacks using the same methods will not make them resist fourth-order, *etc.*

The purpose of this paper is to remedy to these problems. First-order countermeasures have started to be formally studied by Christofi *et al.* [CCGV13], who have been followed by Rauzy & Guilley [RG14a, RG14b], and Barthe *et al.* [BDF+14]. To our best knowledge, no such work has been attempted on high-order countermeasures. Thus, we should understand the working factors of a countermeasure, and use that knowledge to informedly design a generic high-order countermeasure, either one resisting any number of faults, or one which could be customized to protect against n faults, for any given $n \geqslant 1$.

Notice that we consider RSA used in a mode where the BellCoRe attack is applicable; this means that we assume that the attacker can choose (but not necessarily knows) the message that is exponentiated, which is the case in *decryption* mode or in (outdated) *deterministic signature* mode (*e.g.*, PKCS #1 v1.5). In some other modes, formal proofs of security have been conducted [CM09, BDF+14].

Contributions In this paper we propose a classification of the existing CRT-RSA countermeasures against the BellCoRe fault-injection attacks. Doing so, we raise questions whose answers lead to a deeper understanding of how the countermeasures work. We show that the many countermeasures that we study (and their variations) are actually applying a common protection strategy but optimize it in different ways (Sec. 4). We also show that there is no conceptual distinction between test-based and infective countermeasures and how either one can be transformed into the other (Prop. 2). Furthermore, we show that faults on the code (skipping instructions) can be captured by considering only faults on the data (Lem. 1). These intermediate results allow us to improve the state of the art in several ways:

- we fix an existing and that is known to be broken countermeasure (Alg. 10);
- we drastically optimize an existing countermeasure, while at the same time we transform it to be infective instead of test-based (Alg. 11);
- we also show how to upgrade countermeasures to resist any given number of faults: given a correct first-order countermeasure, we present a way to design a provable high-order countermeasure for a well defined and reasonable fault model (Sec. 4.2).

Finally, we pave the way for a generic approach against fault attacks for any modular arithmetic computations, and thus for the automatic insertion of countermeasures.

Organization of the paper We recall the CRT-RSA cryptosystem and the BellCoRe attack in Sec. 2. Then, to better understand the existing countermeasures, we attempt to classify them in Sec. 3, which also presents the state of the art. We then try to capture what make the essence of a countermeasure in Sec. 4, and use that knowledge to determine how to build a high-order countermeasure. We last use our findings to build better countermeasures by fixing and simplifying existing ones in Sec. 5. A discussion about possible attacks that would circumvent the assumptions of our formal model is given in Sec. 6. Conclusions and perspectives are drawn in Sec. 7. The appendices contain the detail of some secondary results.

2 CRT-RSA and the BellCoRe Attack

This section summarizes known results about fault attacks on CRT-RSA (see also [Koç94], [TW12, Chap. 3] and [JT11, Chap. 7 & 8]). Its purpose is to settle the notions and the associated notations that will be used in the later sections, to present our novel contributions.

2.1 RSA

RSA is both an *encryption* and a *signature* scheme. It relies on the fact that for any message $0 \leq M < N$, $(M^d)^e \equiv M \mod N$, where $d \equiv e^{-1} \mod \varphi(N)$, by Euler's theorem[1]. In this equation, φ is Euler's totient function, equal to $\varphi(N) = (p-1) \cdot (q-1)$ when $N = p \cdot q$ is a composite number, product of two primes p and q. For example, if Alice generates the signature $S = M^d \mod N$, then Bob can verify it by computing $S^e \mod N$, which must be equal to M unless Alice is only pretending to know d. Therefore (N, d) is called the private key, and (N, e) the public key. In this paper, we are not concerned about the key generation step of RSA, and simply assume that d is an unknown number in $[\![1, \varphi(N) = (p-1) \cdot (q-1)[\![$. Actually, d can also be chosen to be equal to the smallest value $e^{-1} \mod \lambda(N)$, where $\lambda(N) = \frac{(p-1)\cdot(q-1)}{\gcd(p-1,q-1)}$ is the Carmichael function (see PKCS #1 v2.1, §3.1).

2.2 CRT-RSA

The computation of $M^d \mod N$ can be speeded-up by a factor of four using the Chinese Remainder Theorem (CRT). Indeed, numbers modulo p and q are twice as short as those modulo N. For example, for $2,048$ bits RSA, p and q are $1,024$ bits long. CRT-RSA consists in computing $S_p = M^d \mod p$ and $S_q = M^d \mod q$, which can be recombined into S with a limited overhead. Due to the little Fermat theorem (the special case of the Euler theorem when the modulus is a prime), $S_p = (M \mod p)^{d \mod (p-1)} \mod p$. This means that in the computation of S_p, the processed data have $1,024$ bits, and the exponent itself has $1,024$ bits (instead of $2,048$ bits). Thus the multiplication is four times faster and the exponentiation eight times faster. However, as there are two such exponentiations (modulo p and q), the overall CRT-RSA is roughly speaking four times faster than RSA computed modulo N.

[1] We use the usual convention in all mathematical equations, namely that the "mod" operator has the lowest binding precedence, *i.e.*, $a \times b \mod c \times d$ represents the element $a \times b$ in $\mathbb{Z}_{c \times d}$.

This acceleration justifies that CRT-RSA is always used if the factorization of N as $p \cdot q$ is known. In CRT-RSA, the private key has a richer structure than simply (N, d): it is actually the 5-tuple (p, q, d_p, d_q, i_q), where:

- $d_p \doteq d \mod (p-1)$,
- $d_q \doteq d \mod (q-1)$, and
- $i_q \doteq q^{-1} \mod p$.

The CRT-RSA algorithm is presented in Alg. 1. It is straightforward to check that the signature computed at line 3 belongs to $[\![0, p \cdot q - 1]\!]$. Consequently, no reduction modulo N is necessary before returning S.

Algorithm 1: Unprotected CRT-RSA

Input : Message M, key (p, q, d_p, d_q, i_q)
Output: Signature $M^d \mod N$

1 $S_p = M^{d_p} \mod p$ `// Intermediate signature in` $\mathbb{Z}_p$
2 $S_q = M^{d_q} \mod q$ `// Intermediate signature in` $\mathbb{Z}_q$
3 $S = S_q + q \cdot (i_q \cdot (S_p - S_q) \mod p)$ `// Recombination in` $\mathbb{Z}_N$ `(Garner's method [Gar65])`
4 **return** S

2.3 The BellCoRe Attack

In 1997, a dreadful remark has been made by Boneh, DeMillo and Lipton [BDL97], three staff of Bell Communication Research: Alg. 1 could reveal the secret primes p and q if the line 1 or 2 of the computation is faulted, even in a very random way. The attack can be expressed as the following proposition.

Proposition 1 (BellCoRe attack)**.** *If the intermediate variable S_p (resp. S_q) is returned faulted as $\widehat{S_p}$ (resp. $\widehat{S_q}$)[2], then the attacker gets an erroneous signature $\widehat{S}$, and is able to recover q (resp. p) as* $\gcd(N, S - \widehat{S})$.

Proof. For any integer x, $\gcd(N, x)$ can only take 4 values:

- 1, if N and x are coprime,
- p, if x is a multiple of p,
- q, if x is a multiple of q,
- N, if x is a multiple of both p and q, *i.e.*, of N.

In Alg. 1, if S_p is faulted (*i.e.*, replaced by $\widehat{S_p} \neq S_p$), then $S - \widehat{S} = q \cdot ((i_q \cdot (S_p - S_q) \mod p) - (i_q \cdot (\widehat{S_p} - S_q) \mod p))$, and thus $\gcd(N, S - \widehat{S}) = q$. If S_q is faulted (*i.e.*, replaced by $\widehat{S_q} \neq S_q$), then $S - \widehat{S} \equiv (S_q - \widehat{S_q}) - (q \mod p) \cdot i_q \cdot (S_q - \widehat{S_q}) \equiv 0 \mod p$ because $(q \mod p) \cdot i_q \equiv 1 \mod p$, and thus $S - \widehat{S}$ is a multiple of p. Additionally, $S - \widehat{S}$ is not a multiple of q. So, $\gcd(N, S - \widehat{S}) = p$. □

Before continuing to the next section, we will formalize our attack model by defining what is a fault injection and what is the order of an attack.

[2] In other papers, the faulted variables (such as X) are written either as X^* or $\tilde{X}$; in this paper, we use a hat which can stretch to cover the adequate portion of the expression, as it allows to make an unambiguous difference between $\widehat{X^e}$ and $\widehat{X}^e$.

Definition 1 (Fault injection)**.** During the execution of an algorithm, the attacker can:

- modify any intermediate value by setting it to either a *random value* (*randomizing fault*) or *zero* (*zeroing fault*); such a fault can be either *permanent* (*e.g.*, in memory) or *transient* (*e.g.*, in a register or a bus);
- skip any number of consecutive instructions (*skipping fault*).

At the end of the computation the attacker can read the result returned by the algorithm.

Remark 1. This fault injection model implies that faults can be injected very accurately in timing (the resolution is the clock period), whereas the fault locality in space is poor (the attacker cannot target a specific bit). This models an attacker who is able to identify the sequence of operations by a simple side-channel analysis, but who has no knowledge of the chip internals. Such attack model is realistic for designs where the memories are scrambled and the logic gates randomly routed (in a sea of gates).

Lemma 1. *The effect of a skipping fault (i.e., fault on the code) can be captured by considering only randomizing and zeroing faults (i.e., fault on the data).*

Proof. Indeed, if the skipped instructions are part of an arithmetic operation:

- either the computation has not been done at all and the value in memory where the result is supposed to be stays zero (if initialized) or random (if not),
- or the computation has partly been done and the value written in memory as its result is thus pseudo-randomized (and considered random at our modeling level).

If the skipped instruction is a branching instruction, then it is equivalent to do a zeroing fault on the result of the branching condition to make it false and thus avoid branching. □

Definition 2 (Attack order)**.** We call *order* of the attack the number of fault injections in the computation. An attack is said to be *high-order* if its order is strictly more than 1.

3 Classifying Countermeasures

The goal of a countermeasure against fault-injection attacks is to avoid returning a compromised value to the attacker. To this end, countermeasures attempt to verify the integrity of the computation before returning its result. If the integrity is compromised, then the returned value should be a random number or an error constant, in order not to leak any information.

An obvious way of achieving that goal is to repeat the computation and compare the results, but this approach is very expensive in terms of computation time. The same remark applies to the verification of the signature (notice that e can be recovered for this purpose from the 5-tuple (p, q, d_p, d_q, i_q), as explained in App. A). In this section we explore the different methods used by the existing countermeasures to verify the computation integrity faster than $(M^d)^e \stackrel{?}{\equiv} M \mod N$. Besides, we recall that such a verification is prone to other attacks [YJ00], and must thus be avoided.

3.1 Shamir's or Giraud's Family of Countermeasures

To the authors knowledge, there are two main families of countermeasures: those which are descendants of Shamir's countermeasure [Sha99], and those which are descendants of Giraud's [Gir06].

The countermeasures in Giraud's family avoid replicating the computations using particular exponentiation algorithms. These algorithms keep track of variables involved in intermediate steps; those help verifying the consistency of the final results by a consistency check of an invariant that is supposed to be spread till the last steps. This idea is illustrated in Alg. 2, which resembles the one of Giraud. The test at line 5 verifies that the recombined values S and S' (recombination of intermediate steps of the exponentiation) are consistent. Example of other countermeasures in this family are the ones of Boscher *et al.* [BNP07], Rivain [Riv09] (and its recently improved version [LRT14]), or Kim *et al.* [KKHH11]. The former two mainly optimize Giraud's, while the latter introduce an infective verification based on binary masks. The detailed study of the countermeasures in Giraud's family is left as future work.

Algorithm 2: CRT-RSA with a Giraud's family countermeasure

Input : Message M, key (p, q, d_p, d_q, i_q)
Output: Signature $M^d \mod N$, or error

1 $(S_p, S'_p) = \mathsf{ExpAlgorithm}(M, d_p)$ `// ExpAlgorithm(a, b) returns` (a^b, a^{b-1})
2 $(S_q, S'_q) = \mathsf{ExpAlgorithm}(M, d_q)$
3 $S = S_q + q \cdot (i_q \cdot (S_p - S_q) \mod p)$ `// Recombination`
4 $S' = S'_q + q \cdot (i_q \cdot (S'_p - S'_q) \mod p)$ `// Recombination for verification`
5 **if** $M \cdot S' \not\equiv S \mod pq$ **then return** error
6 **return** S

Indeed, the rest of our paper is mainly concerned with Shamir's family of countermeasures. The countermeasures in Shamir's family rely on a kind of "checksum" of the computation using smaller numbers (the checksum is computed in rings smaller than the ones of the actual computation). The base-two logarithm of the smaller rings cardinal is typically equal to 32 or to 64 (bits): therefore, assuming that the faults are randomly distributed, the probability of having an undetected fault is 2^{-32} or 2^{-64}, *i.e.*, very low. In the sequel, we will make a language abuse by considering that such probability is equal to zero. We also use the following terminology:

Notation 1. *Let a a big number and b a small number, such that they are coprime. We call the ring $\mathbb{Z}_{ab}$ an* overring *of $\mathbb{Z}_a$, and the ring $\mathbb{Z}_b$ a* subring *of $\mathbb{Z}_{ab}$.*

Remark 2. RSA is friendly to protections by checksums because it computes in rings $\mathbb{Z}_a$ where a is either a large prime number (*e.g.*, $a = p$ or $a = q$) or the product of large prime numbers (*e.g.*, $a = p \cdot q$). Thus, any small number $b > 1$ is coprime with a, and so we have an isomorphism between the *overring* $\mathbb{Z}_{ab}$ and the direct product of $\mathbb{Z}_a$ and $\mathbb{Z}_b$, *i.e.*, $\mathbb{Z}_{ab} \cong \mathbb{Z}_a \times \mathbb{Z}_b$. This means that the Chinese Remainder Theorem applies. Consequently, the nominal computation and the checksum can be conducted in parallel in $\mathbb{Z}_{ab}$.

The countermeasures attempt to assert that some invariants on the computations and the checksums hold. There are many different ways to use the checksums and to verify these invariants. In the rest of this section we review these ways while we attempt to classify countermeasures and understand better what are the necessary invariants to verify.

3.2 Test-Based or Infective

A first way to classify countermeasures is to separate those which consist in step-wise internal checks during the CRT computation and those which use an infective computation strategy to make the result unusable by the attacker in case of fault injection.

Definition 3 (Test-based countermeasure)**.** A countermeasure is said to be *test-based* if it attempts to detect fault injections by verifying that some arithmetic invariants are respected, and branch to return an error instead of the numerical result of the algorithm in case of invariant violation. Examples of test-based countermeasures are the ones of Shamir [Sha99], Aumüller *et al.* [ABF+02], Vigilant [Vig08a], or Joye *et al.* [JPY01].

Definition 4 (Infective countermeasure)**.** A countermeasure is said to be *infective* if rather than testing arithmetic invariants it uses them to compute a neutral element of some arithmetic operation in a way that would not result in this neutral element if the invariant is violated. It then uses the results of these computations to infect the result of the algorithm before returning it to make it unusable by the attacker (thus, it does not need branching instructions). Examples of infective countermeasures are the ones by Blömer *et al.* [BOS03] (and the variant by Liu *et al.* [LKW06]), Ciet & Joye [CJ05], or Kim *et al.* [KKHH11].

The extreme similarity between the verifications in the test-based countermeasure of Joye *et al.* [JPY01] (see Alg. 3, line 9) and the infective countermeasure of Ciet & Joye [CJ05] (see Alg. 4, lines 10 and 11) is striking, but it is actually not surprising at all, as we will discover in Prop. 2.

Algorithm 3: CRT-RSA with Joye *et al.*'s countermeasure [JPY01]

Input : Message M, key (p, q, d_p, d_q, i_q)
Output: Signature $M^d \mod N$, or error

1 Choose two small random integers r_1 and r_2.
2 Store in memory $p' = p \cdot r_1$, $q' = q \cdot r_2$, $i'_q = q'^{-1} \mod p'$, $N = p \cdot q$.
3 $S'_p = M^{d_p \mod \varphi(p')} \mod p'$ // Intermediate signature in $\mathbb{Z}_{pr_1}$
4 $S_{pr} = M^{d_p \mod \varphi(r_1)} \mod r_1$ // Checksum in $\mathbb{Z}_{r_1}$
5 $S'_q = M^{d_q \mod \varphi(q')} \mod q'$ // Intermediate signature in $\mathbb{Z}_{qr_2}$
6 $S_{qr} = M^{d_q \mod \varphi(r_2)} \mod r_2$ // Checksum in $\mathbb{Z}_{r_2}$
7 $S_p = S'_p \mod p$ // Retrieve intermediate signature in $\mathbb{Z}_p$
8 $S_q = S'_q \mod q$ // Retrieve intermediate signature in $\mathbb{Z}_q$
9 **if** $S'_p \not\equiv S_{pr} \mod r_1$ **or** $S'_q \not\equiv S_{qr} \mod r_2$ **then return** error
10 **return** $S = S_q + q \cdot (i_q \cdot (S_p - S_q) \mod p)$ // Recombination in $\mathbb{Z}_N$

Proposition 2 (Equivalence between test-based and infective countermeasures)**.** *Each test-based (resp. infective) countermeasure has a direct equivalent infective (resp. test-based) countermeasure.*

Proof. The invariants that must be verified by countermeasures are modular equality, so they are of the form $a \stackrel{?}{\equiv} b \mod m$, where a, b and m are arithmetic expressions.

It is straightforward to transform this invariant into a Boolean expression usable in test-based countermeasures: `if a != b [mod m] then return` *error*.

Algorithm 4: CRT-RSA with Ciet & Joye's countermeasure [CJ05]

Input : Message M, key (p, q, d_p, d_q, i_q)
Output: Signature $M^d \mod N$, or a random value in $\mathbb{Z}_N$

1 Choose small random integers r_1, r_2, and r_3.
2 Choose a random integer a.
3 Initialize γ with a random number
4 Store in memory $p' = p \cdot r_1$, $q' = q \cdot r_2$, $i'_q = q'^{-1} \mod p'$, $N = p \cdot q$.
5 $S'_p = a + M^{d_p \mod \varphi(p')} \mod p'$ // Intermediate signature in $\mathbb{Z}_{pr_1}$
6 $S_{pr} = a + M^{d_p \mod \varphi(r_1)} \mod r_1$ // Checksum in $\mathbb{Z}_{r_1}$
7 $S'_q = a + M^{d_q \mod \varphi(q')} \mod q'$ // Intermediate signature in $\mathbb{Z}_{qr_2}$
8 $S_{qr} = a + M^{d_q \mod \varphi(r_2)} \mod r_2$ // Checksum in $\mathbb{Z}_{r_2}$
9 $S' = S'_q + q' \cdot (i'_q \cdot (S'_p - S'_q) \mod p')$ // Recombination in $\mathbb{Z}_{Nr_1r_2}$
10 $c_1 = S' - S_{pr} + 1 \mod r_1$ // Invariant for the signature modulo p
11 $c_2 = S' - S_{qr} + 1 \mod r_2$ // Invariant for the signature modulo q
12 $\gamma = (r_3 \cdot c_1 + (2^l - r_3) \cdot c_2)/2^l$ // $\gamma = 1$ if c_1 and c_2 have value 1
13 **return** $S = S' - a^\gamma \mod N$ // Infection and result retrieval in $\mathbb{Z}_N$

To use it in infective countermeasures, it is as easy to verify the same invariant by computing a value which should be 1 if the invariant holds: `c := a - b + 1 mod m`. The numbers obtained this way for each invariant can then be multiplied and their product c^*, which is 1 only if all invariants are respected, can be used as an exponent on the algorithm's result to infect it if one or more of the tested invariants are violated. Indeed, when the attacker perform the BellCoRe attack by computing $\gcd(N, S - \widehat{S^{c^*}})$ as defined in Prop. 1, then if c^* is not 1 the attack would not work. □

By Prop. 2, we know that there is an equivalence between test-based and infective countermeasures. This means that in theory any attack working on one kind of countermeasure will be possible on the equivalent countermeasure of the other kind. However, we remark that in practice it is harder to do a zeroing fault on an intermediate value (especially if it is the result of a computation with big numbers) in the case of an infective countermeasure, than it is to skip one branching instruction in the case of a test-based countermeasure. We conclude from this the following rule of thumb: *it is better to use the infective variant of a countermeasure.* In addition, it is generally the case that code without branches is safer (think of timing attacks or branch predictor attacks on modern CPUs).

Note that if a fault occurs, c^* is not 1 anymore and thus the computation time required to compute S^{c^*} might significantly increase. This is not a security problem, indeed, taking longer to return a randomized value in case of an attack is not different from rapidly returning an error constant without finishing the computation first as it is done in the existing test-based countermeasures. In the worst case scenario, the additional time would be correlated to the induced fault, but we assume the fault to be controlled by the attacker already.

3.3 Intended Order

Countermeasures can be classified depending on their order, *i.e.*, the maximum order of the attacks (as per Def. 2) that they can protect against.

In the literature concerning CRT-RSA countermeasures against fault-injection attacks, most countermeasures claim to be first-order, and a few claim second-order resistance. For instance, the countermeasures by Aumüller *et al.* [ABF+02] and the one by Vigilant [Vig08a] are described as first-order by their authors, while Ciet & Joye [CJ05] describe a second-order fault model and propose a countermeasure which is supposed to resist to this fault model, and thus be second-order.

However, using the finja[3] tool which has been open-sourced by Rauzy & Guilley [RG14a], we found out that the countermeasure of Ciet & Joye is in fact vulnerable to second-order attacks (in our fault model of Def. 1). This is not very surprising. Indeed, Prop. 2 proves that injecting a fault, and then skipping the invariant verification which was supposed to catch the first fault injection, is a second-order attack strategy which also works for infective countermeasures, except the branching-instruction skip has to be replaced by a zeroing fault. As expected, the attacks we found using finja did exactly that. For instance a zeroing fault on S'_p (resp. S'_q) makes the computation vulnerable to the BellCoRe attack, and a following zeroing fault on S_{pr} (resp. S_{qr}) makes the verification pass anyway. To our knowledge our attack is new. It is indeed different from the one Dottax *et al.* [DGRS09] found and fixed in their paper, which was an attack on the use of γ (see line 12 of Alg. 4). It is true that their attack model only allows skipping faults (as per Def. 1) for the second injection, but we have concerns about this:

- What justifies this limitation on the second fault? Surely if the attackers are able to inject two faults and can inject a zeroing fault once they can do it twice.
- Even considering their attack model, a zeroing fault on an intermediate variable x can in many cases be obtained by skipping the instructions where the writing to x happens.
- The fixed version of the countermeasure by Dottax *et al.* [DGRS09, Alg. 8, p. 13] makes it even closer to the one of Joye *et al.* by removing the use of a and γ. It also removes the result infection part and instead returns S along with values that should be equal if no faults were injected, leaving "out" of the algorithm the necessary comparison and branching instructions which are presented in a separate procedure [DGRS09, Proc. 1, p. 11]. The resulting countermeasure is second-order resistant (in their attack model) only because the separate procedure does the necessary tests twice (it would indeed break at third-order unless an additional repetition of the test is added, *etc.*).

An additional remark would be that the algorithms of intended second-order countermeasures does not look very different from others. Moreover, Rauzy & Guilley [RG14a, RG14b] exposed evidence that the intendedly first-order countermeasures of Aumüller *et al.* and Vigilant actually offer the same level of resistance against second-order attacks, *i.e.*, they resist when the second injected fault is a randomizing fault (or a skipping fault which amounts to a randomizing fault).

3.4 Usage of the Small Rings

In most countermeasures, the computation of the two intermediate signatures modulo p and modulo q of the CRT actually takes place in overrings. The computation of S_p (resp. S_q) is done in $\mathbb{Z}_{pr_1}$ (resp. $\mathbb{Z}_{qr_2}$) for some small random number r_1 (resp. r_2) rather than in $\mathbb{Z}_p$ (resp. $\mathbb{Z}_q$). This allows

[3] `http://pablo.rauzy.name/sensi/finja.html` (we used the commit `782384a` version of the code).

the retrieval of the results by reducing modulo p (resp. q) and verifying the signature modulo r_1 (resp. r_2), or, if it is done after the CRT recombination, the results can be retrieved by reducing modulo $N = p \cdot q$. The reduction in the *small subrings* $\mathbb{Z}_{r_1}$ and $\mathbb{Z}_{r_2}$ is used as the checksums for verifying the integrity of the computation. It works because small random numbers are necessarily coprime with a big prime number.

An interesting part of countermeasures is how they use the small subrings to verify the integrity of the computations. Almost all the various countermeasures we studied had different ways of using them. However, they can be divided in two groups. On one side there are countermeasures which use the small subrings to verify the integrity of the intermediate CRT signatures and of the recombination directly but using smaller numbers, like Blömer *et al.*'s countermeasure [BOS03], or Ciet & Joye's one [CJ05]. On the other side, there are countermeasures which use some additional arithmetic properties to verify the necessary invariants indirectly in the small subrings. Contrary to the countermeasures in the first group, the ones in the second group use the same value r for r_1 and r_2. The symmetry obtained with $r_1 = r_2$ is what makes the additional arithmetic properties hold, as we will see.

3.4.1 Verification of the Intermediate CRT Signatures

The countermeasure of Blömer *et al.* [BOS03] uses the small subrings to verify the intermediate CRT signatures. It is exposed in Alg. 5. This countermeasure needs access to d directly rather than d_p and d_q as the standard interface for CRT-RSA suggests, in order to compute $d'_p = d \mod \varphi(p \cdot r_1)$ and $d'_q = d \mod \varphi(q \cdot r_2)$, as well as their inverse $e'_p = {d'_p}^{-1} \mod \varphi(p \cdot r_1)$ and $e'_q = {d'_q}^{-1} \mod \varphi(q \cdot r_2)$ to verify the intermediate CRT signatures.

We can see in Alg. 5 that these verifications (lines 6 and 7) happen after the recombination (line 5) and retrieve the checksums in $\mathbb{Z}_{r_1}$ (for the p part of the CRT) and $\mathbb{Z}_{r_2}$ (for the q part) from the recombined value S'. It allows these tests to verify the integrity of the recombination at the same time as they verify the integrity of the intermediate CRT signatures.

Algorithm 5: CRT-RSA with Blömer *et al.*'s countermeasure [BOS03]

Input : Message M, key (p, q, d, i_q)
Output: Signature $M^d \mod N$, or a random value in $\mathbb{Z}_N$

1 Choose two small random integers r_1 and r_2.
2 Store in memory $p' = p \cdot r_1$, $q' = q \cdot r_2$, $i'_q = {q'}^{-1} \mod p'$, $N = p \cdot q$, $N' = N \cdot r_1 \cdot r_2$, d'_p, d'_q, e'_p, e'_q.
3 $S'_p = M^{d'_p} \mod p'$ // Intermediate signature in $\mathbb{Z}_{pr_1}$
4 $S'_q = M^{d'_q} \mod q'$ // Intermediate signature in $\mathbb{Z}_{qr_2}$
5 $S' = S'_q + q' \cdot (i'_q \cdot (S'_p - S'_q) \mod p')$ // Recombination in $\mathbb{Z}_{Nr_1r_2}$
6 $c_1 = M - {S'}^{e'_p} + 1 \mod r_1$ // Invariant for the signature modulo p
7 $c_2 = M - {S'}^{e'_q} + 1 \mod r_2$ // Invariant for the signature modulo q
8 **return** $S = {S'}^{c_1 c_2} \mod N$ // Infection and result retrieval in $\mathbb{Z}_N$

3.4.2 Checksums of the Intermediate CRT Signatures

The countermeasure of Ciet & Joye [CJ05] uses the small subrings to compute checksums of the intermediate CRT signatures. It is exposed in Alg. 4. Just as the previous one, the verifications (lines 10 and 11) take place after the recombination (line 9) and retrieve the checksums in $\mathbb{Z}_{r_1}$ (for the p part of the CRT) and $\mathbb{Z}_{r_2}$ (for the q part) from the recombined value S', which enables the integrity verification of the recombination at the same time as the integrity verifications of the intermediate CRT signatures.

We note that this is missing from the protection of Joye *et al.* [JPY01], presented in Alg. 3, which does not verify the integrity of the recombination at all and is thus as broken as Shamir's countermeasure [Sha99]. The countermeasure of Ciet & Joye is a clever fix against the possible fault attacks on the recombination of Joye *et al.*'s countermeasure, which also uses the transformation that we described in Prop. 2 from a test-based to an infective countermeasure.

3.4.3 Overrings for CRT Recombination

In Ciet & Joye's countermeasure the CRT recombination happens in an overring $\mathbb{Z}_{Nr_1r_2}$ of $\mathbb{Z}_N$ while Joye *et al.*'s countermeasure extracts in $\mathbb{Z}_p$ and $\mathbb{Z}_q$ the results S_p and S_q of the intermediate CRT signatures to do the recombination in $\mathbb{Z}_N$ directly.

There are only two other countermeasures which do the recombination in $\mathbb{Z}_N$ that we know of: the one of Shamir [Sha99] and the one of Aumüller *et al.* [ABF$^+$02]. The first one is known to be broken, in particular because it does not check whether the recombination has been faulted at all. The second one seems to need to verify 5 invariants to resist the BellCoRe attack[4], which is more than the only 2 required by the countermeasure of Ciet & Joye [CJ05] or by the one of Blömer *et al.* [BOS03], while offering a similar level of protection (see [RG14a]). This fact led us to think that the additional tests are necessary because the recombination takes place "in the clear". But we did not jump right away to that conclusion. Indeed, Vigilant's countermeasure [Vig08a] does the CRT recombination in the $\mathbb{Z}_{Nr^2}$ overring of $\mathbb{Z}_N$ and seems to require 7 verifications[5] to also offer that same level of security (see [RG14b]). However, we remark that Shamir's, Aumüller *et al.*'s, and Vigilant's countermeasures use the same value for r_1 and r_2.

3.4.4 Identity of r_1 and r_2

Some countermeasures, such as the ones of Shamir [Sha99], Aumüller *et al.* [ABF$^+$02], and Vigilant [Vig08a] use a single random number r to construct the overrings used for the two intermediate CRT signatures computation. The resulting symmetry allows these countermeasures to take advantage of some additional arithmetic properties.

Shamir's countermeasure In his countermeasure, which is presented in Alg. 6, Shamir uses a clever invariant property to verify the integrity of both intermediate CRT signatures in a single verification step (line 9). This is made possible by the fact that he uses d directly instead of d_p and d_q, and thus the checksums in $\mathbb{Z}_r$ of both the intermediate CRT signatures are supposed to be

[4] The original Aumüller *et al.*'s countermeasure uses 7 verifications because it also needs to check the integrity of intermediate values introduced against simple power analysis, see [RG14a, Remark 1].

[5] Vigilant's original countermeasure and its corrected version by Coron *et al.* [CGM$^+$10] actually use 9 verifications but were simplified by Rauzy & Guilley [RG14b] who removed 2 verifications.

equal if no fault occurred. Unfortunately, the integrity of the recombination is not verified at all. We will see in Sec. 5.1 how to fix this omission. Besides, we notice that d can be reconstructed from a usual CRT-RSA key (p, q, d_p, d_q, i_q); we refer the reader to Appendix A.

Algorithm 6: CRT-RSA with Shamir's countermeasure [Sha99]

Input : Message M, key (p, q, d, i_q)
Output: Signature $M^d \mod N$, or error

1 Choose a small random integer r.
2 $p' = p \cdot r$
3 $S'_p = M^{d \mod \varphi(p')} \mod p'$ // Intermediate signature in $\mathbb{Z}_{pr}$
4 $q' = q \cdot r$
5 $S'_q = M^{d \mod \varphi(q')} \mod q'$ // Intermediate signature in $\mathbb{Z}_{qr}$
6 $S_p = S'_p \mod p$ // Retrieve intermediate signature in $\mathbb{Z}_p$
7 $S_q = S'_q \mod q$ // Retrieve intermediate signature in $\mathbb{Z}_q$
8 $S = S_q + q \cdot (i_q \cdot (S_p - S_q) \mod p)$ // Recombination in $\mathbb{Z}_N$
9 **if** $S'_p \not\equiv S'_q \mod r$ **then return** error
10 **return** S

Aumüller *et al.*'s countermeasure Contrary to Shamir, Aumüller *et al.* do verify the integrity of the recombination in their countermeasure, which is presented in Alg. 7. To do this, they straightforwardly check (line 10) that when reducing the result S of the recombination modulo p (resp. q), the obtained value corresponds to the intermediate signature in $\mathbb{Z}_p$ (resp. $\mathbb{Z}_q$). However, they do not use d directly but rather conform to the standard CRT-RSA interface by using d_p and d_q. Thus, they need another verification to check the integrity of the intermediate CRT signatures. Their clever strategy is to verify that the checksums of S_p and S_q in $\mathbb{Z}_r$ are conform to each other (lines 11 to 13). For that they check whether ${S_p}^{d_q}$ is equal to ${S_q}^{d_p}$ in $\mathbb{Z}_r$, that is, whether the invariant $(M^{d_p})^{d_q} \equiv (M^{d_q})^{d_p} \mod r$ holds.

The two additional tests on line 4 verify the integrity of p' and q'. Indeed, if p or q happen to be randomized when computing p' or q' the invariant verifications in $\mathbb{Z}_r$ would pass but the retrieval of the intermediate signatures in $\mathbb{Z}_p$ or $\mathbb{Z}_q$ would return random values, which would make the BellCoRe attack work. These important verifications are missing from all the previous countermeasures in Shamir's family.

Vigilant's countermeasure Vigilant takes another approach. Rather than doing the integrity verifications on "direct checksums" that are the representative values of the CRT-RSA computation in the small subrings, Vigilant uses different values that he constructs for that purpose. The clever idea of his countermeasure is to use sub-CRTs on the values that the CRT-RSA algorithm manipulates in order to have in one part the value we are interested in and in the other the value constructed for the verification (lines 8 and 17).

Algorithm 7: CRT-RSA with Aumüller *et al.*'s countermeasure[6] [ABF+02]

Input : Message M, key (p, q, d_p, d_q, i_q)
Output: Signature $M^d \mod N$, or error

1 Choose a small random integer r.
2 $p' = p \cdot r$
3 $q' = q \cdot r$
4 **if** $p' \not\equiv 0 \mod p$ **or** $q' \not\equiv 0 \mod q$ **then return** error
5 $S'_p = M^{d_p \mod \varphi(p')} \mod p'$ // Intermediate signature in $\mathbb{Z}_{pr}$
6 $S'_q = M^{d_q \mod \varphi(q')} \mod q'$ // Intermediate signature in $\mathbb{Z}_{qr}$
7 $S_p = S'_p \mod p$ // Retrieve intermediate signature in $\mathbb{Z}_p$
8 $S_q = S'_q \mod q$ // Retrieve intermediate signature in $\mathbb{Z}_q$
9 $S = S_q + q \cdot (i_q \cdot (S_p - S_q) \mod p)$ // Recombination in $\mathbb{Z}_N$
10 **if** $S \not\equiv S'_p \mod p$ **or** $S \not\equiv S'_q \mod q$ **then return** error
11 $S_{pr} = S'_p \mod r$ // Checksum of S_p in $\mathbb{Z}_r$
12 $S_{qr} = S'_q \mod r$ // Checksum of S_q in $\mathbb{Z}_r$
13 **if** ${S_{pr}}^{d_q \mod \varphi(r)} \not\equiv {S_{qr}}^{d_p \mod \varphi(r)} \mod r$ **then return** error
14 **return** S

To do this, he transforms M into another value M' such that:

$$M' \equiv \begin{cases} M & \mod N, \\ 1 + r & \mod r^2, \end{cases}$$

which implies that:

$$S' = M'^d \mod Nr^2 \equiv \begin{cases} M^d & \mod N, \\ 1 + dr & \mod r^2. \end{cases}$$

The latter results are based on the binomial theorem, which states that $(1+r)^d = \sum_{k=0}^{d} \binom{d}{k} r^k = 1 + dr + \binom{d}{2} r^2 + \ldots$, which simplifies to $1 + dr$ in the $\mathbb{Z}_{r^2}$ ring.

This property is used to verify the integrity of the intermediate CRT signatures on lines 11 and 20. It is also used on line 24 which tests the recombination using the same technique but with random values inserted on lines 21 and 22 in place of the constructed ones. This test also verifies the integrity of N.

Two additional tests are required by Vigilant's arithmetic trick. The verifications at lines 10 and 19 ensure that the original message M has indeed been CRT-embedded in M'_p and M'_q.

4 The Essence of a Countermeasure

Our attempt to classify the existing countermeasures provided us with a deep understanding of how they work. To ensure the integrity of the CRT-RSA computation, the algorithm must verify

[6] For the sake of simplicity we removed some code that served against SPA (simple power analysis) and only kept the necessary code against fault-injection attacks.

Algorithm 8: CRT-RSA with Vigilant's countermeasure[6] [Vig08a]
with Coron *et al.*'s fixes [CGM+10] and Rauzy & Guilley's simplifications [RG14b]

Input : Message M, key (p, q, d_p, d_q, i_q)
Output: Signature $M^d \mod N$, or error

1 Choose small random integers r, R_1, and R_2.
2 $N = p \cdot q$
3 $p' = p \cdot r^2$
4 $i_{pr} = p^{-1} \mod r^2$
5 $M_p = M \mod p'$
6 $B_p = p \cdot i_{pr}$
7 $A_p = 1 - B_p \mod p'$
8 $M'_p = A_p \cdot M_p + B_p \cdot (1 + r) \mod p'$ // CRT insertion of verification value in M'_p
9 $S'_p = {M'_p}^{d_p \mod \varphi(p')} \mod p'$ // Intermediate signature in $\mathbb{Z}_{pr^2}$
10 **if** $M'_p \not\equiv M \mod p$ **then return** error
11 **if** $B_p \cdot S'_p \not\equiv B_p \cdot (1 + d_p \cdot r) \mod p'$ **then return** error
12 $q' = q \cdot r^2$
13 $i_{qr} = q^{-1} \mod r^2$
14 $M_q = M \mod q'$
15 $B_q = q \cdot i_{qr}$
16 $A_q = 1 - B_q \mod q'$
17 $M'_q = A_q \cdot M_q + B_q \cdot (1 + r) \mod q'$ // CRT insertion of verification value in M'_q
18 $S'_q = {M'_q}^{d_q \mod \varphi(q')} \mod q'$ // Intermediate signature in $\mathbb{Z}_{qr^2}$
19 **if** $M'_q \not\equiv M \mod q$ **then return** error
20 **if** $B_q \cdot S'_q \not\equiv B_q \cdot (1 + d_q \cdot r) \mod q'$ **then return** error
21 $S_{pr} = S'_p - B_p \cdot (1 + d_p \cdot r - R_1)$ // Verification value of S'_p swapped with R_1
22 $S_{qr} = S'_q - B_q \cdot (1 + d_q \cdot r - R_2)$ // Verification value of S'_q swapped with R_2
23 $S_r = S_{qr} + q \cdot (i_q \cdot (S_{pr} - S_{qr}) \mod p')$ // Recombination in $\mathbb{Z}_{Nr^2}$
// Simultaneous verification of lines 2 and 23
24 **if** $pq \cdot (S_r - R_2 - q \cdot i_q \cdot (R_1 - R_2)) \not\equiv 0 \mod Nr^2$ **then return** error
25 **return** $S = S_r \mod N$ // Retrieve result in $\mathbb{Z}_N$

3 things: the integrity of the computation modulo p, the integrity of the computation modulo q, and the integrity of the CRT recombination (which can be subject to transient fault attacks). This fact has been known since the first attacks on Shamir's countermeasure. Our study of the existing countermeasures revealed that, as expected, those which perform these three integrity verifications are the ones which actually work. This applies to Shamir's family of countermeasures, but also for Giraud's family. Indeed, countermeasures in the latter also verify the two exponentiations and the recombination by testing the consistency of the exponentiations indirectly on the recombined value.

4.1 A Straightforward Countermeasure

The result of these observations is a very straightforward countermeasure, presented in Alg. 9. This countermeasure works by testing the integrity of the signatures modulo p and q by replicating the computations (lines 1 and 3) and comparing the results, and the integrity of the recombination by verifying that the two parts of the CRT can be retrieved from the final result (line 5). This countermeasure is of course very expensive since the two big exponentiations are done twice, and is thus not usable in practice. Note that it is nonetheless still better in terms of speed than computing RSA without the CRT optimization.

Algorithm 9: CRT-RSA with straightforward countermeasure

Input : Message M, key (p, q, d_p, d_q, i_q)
Output: Signature $M^d \mod N$, or error

1 $S_p = M^{d_p \mod \varphi(p)} \mod p$ // Intermediate signature in $\mathbb{Z}_p$
2 **if** $S_p \not\equiv M^{d_p} \mod p$ **then return** error
3 $S_q = M^{d_q \mod \varphi(q)} \mod q$ // Intermediate signature in $\mathbb{Z}_q$
4 **if** $S_q \not\equiv M^{d_q} \mod q$ **then return** error
5 $S = S_q + q \cdot (i_q \cdot (S_p - S_q) \mod p)$ // Recombination in $\mathbb{Z}_N$
6 **if** $S \not\equiv S_p \mod p$ **or** $S \not\equiv S_q \mod q$ **then return** error
7 **return** S

Proposition 3 (Correctness)**.** *The straightforward countermeasure (and thus all the ones which do equivalent verifications) is secure against first-order fault attacks as per Def. 1 and 2.*

Proof. The proof is in two steps. First, prove that if the intermediate signatures are not correct, then the tests at lines 2 and 4 returns error. Second, prove that if both tests passed then either the recombination is correct or the test at line 6 returns error.

If a fault occurs during the computation of S_p (line 1), then it either has the effect of zeroing its value or randomizing it, as shown by Lem. 1. Thus, the test of line 2 detects it since the two compared values won't be equal. If the fault happens on line 2, then either we are in a symmetrical case: the repeated computation is faulted, or the test is skipped: in that case there are no faults affecting the data so the test is unnecessary anyway. It works similarly for the intermediate signature in $\mathbb{Z}_q$.

If the first two tests pass, then the tests at line 6 verify that both parts of the CRT computation are indeed correctly recombined in S. If a fault occurs during the recombination on line 5 it will

thus be detected. If the fault happens at line 6, then either it is a fault on the data and one of the two tests returns error, or it is a skipping fault which bypasses one or both tests but in that case there are no faults affecting the data so the tests are unnecessary anyway. □

4.2 High-Order Countermeasures

Using the finja[3] tool we were able to verify that removing one of the three integrity checks indeed breaks the countermeasure against first-order attacks. Nonetheless, each countermeasure which has these three integrity checks, plus those that may be necessary to protect optimizations on them, offers the same level of protection.

Proposition 4 (High-order countermeasures)**.** *Against randomizing faults, all correct countermeasures (as per Prop. 3) are high-order. However, there are no generic high-order countermeasures if the three types of faults in our attack model are taken into account, but it is possible to build nth-order countermeasures for any n.*

Proof. Indeed, if a countermeasure is able to detect a single randomizing fault, then adding more faults will not break the countermeasure, since a random fault cannot induce a verification skip. Thus, *all working countermeasures are high-order against randomizing faults.*

However, if after one or more faults which permit an attack, there is a skipping fault or a zeroing fault which leads to skip the verification which would detect the previous fault injections, then the attack will work. As Lem. 1 and Prop. 2 explain, this is true for all countermeasures, not only those which are test-based but also the infective ones. It seems that the only way to protect against that is to replicate of the integrity checks. If each invariant is verified n times, then the countermeasure will resist at least n faults in the worst case scenario: a single fault is used to break the computation and the n others to avoid the verifications which detect the effect of the first fault. Thus, *there are no generic high-order countermeasures* if the three types of faults in our attack model are taken into account, ***but it is possible to build a*** n**th-order countermeasure for any** n **by replicating the invariant verifications** n **times**. □

Existing first-order countermeasures such as the ones of Aumüller *et al.* (Alg. 7, 13), Vigilant (Alg. 8, 11), or Ciet & Joye (Alg. 4) can thus be transformed into nth-order countermeasures, in the attack model described in Def. 1 and 2. As explained, the transformation consists in replicating the verifications n times, whether they are test-based or infective.

This result means that it is very important that the verifications be cost effective. Fortunately, as we saw in Sec. 3 and particularly in Sec. 3.4 on the usage of the small rings, the existing countermeasures offer exactly that: optimized versions of Alg. 9 that use a variety of invariant properties to avoid replicating the two big exponentiations of the CRT computation.

5 Building Better or Different Countermeasures

In the two previous sections we learned a lot about current countermeasures and how they work. We saw that to reduce their cost, most countermeasures use invariant properties to optimize the verification speed by using checksums on smaller numbers than the big ones which are manipulated by the protected algorithm. Doing so, we understood how these optimizations work and the power of their underlying ideas. In this section apply our newly acquired knowledge on the *essence of*

countermeasures in order to build the *quintessence of countermeasures*. Namely, we leverage our findings to fix Shamir's countermeasure, and to drastically simplify the one of Vigilant, while at the same time transforming it to be infective instead of test-based.

5.1 Correcting Shamir's Countermeasure

We saw that Shamir's countermeasure is broken in multiple ways, which has been known for a long time now. To fix it without denaturing it, we need to verify the integrity of the recombination as well as the ones of the overrings moduli. We can directly take these verifications from Aumüller *et al.*'s countermeasure. The result can be observed in Alg. 10.

Algorithm 10: CRT-RSA with a fixed version of Shamir's countermeasure
(new algorithm contributed in this paper)

Input : Message M, key (p, q, d, i_q)
Output: Signature $M^d \mod N$, or error

1 Choose a small random integer r.
2 $p' = p \cdot r$
3 $q' = q \cdot r$
4 **if** $p' \not\equiv 0 \mod p$ **or** $q' \not\equiv 0 \mod q$ **then return** error
5 $S'_p = M^{d \mod \varphi(p')} \mod p'$ // Intermediate signature in $\mathbb{Z}_{pr}$
6 $S'_q = M^{d \mod \varphi(q')} \mod q'$ // Intermediate signature in $\mathbb{Z}_{qr}$
7 **if** $S'_p \not\equiv S'_q \mod r$ **then return** error
8 $S_p = S'_p \mod p$ // Retrieve intermediate signature in $\mathbb{Z}_p$
9 $S_q = S'_q \mod q$ // Retrieve intermediate signature in $\mathbb{Z}_q$
10 $S = S_q + q \cdot (i_q \cdot (S_p - S_q) \mod p)$ // Recombination in $\mathbb{Z}_N$
11 **if** $S \not\equiv S'_p \mod p$ **or** $S \not\equiv S'_q \mod q$ **then return** error
12 **return** S

The additional tests on line 4 protect against transient faults on p (resp. q) while computing p' (resp. q'), which would amount to a randomization of S'_p (resp. S'_q) while computing the intermediate signatures. The additional test on line 7 verifies the integrity of the intermediate signature computations.

5.2 Simplifying Vigilant's Countermeasure

The mathematical tricks used in the Vigilant countermeasure are very powerful. Their understanding enabled the optimization of his countermeasure to only need 3 verifications, while the original version has 9. Our simplified version of the countermeasure can be seen in Alg. 11. Our idea is that it is not necessary to perform the checksum value replacements at lines 21 and 22 of Alg. 8 (see Sec. 3.4). What is more, if these replacements are not done, then the algorithm's computations carry the CRT-embedded checksum values until the end, and *the integrity of the whole computation can be tested with a single verification* in $\mathbb{Z}_{r^2}$ (line 23 of Alg. 11).

This idea not only reduces the number of required verifications, which is in itself a security improvement as shown in Sec. 3.2, but it also optimizes the countermeasure for speed and reduces

its need for randomness (the computations of lines 21 and 22 of Alg. 8 are removed).

The two other tests that are left are the ones of lines 10 and 19 in Alg. 8, which ensure that the original message M has indeed been CRT-embedded in M'_p and M'_q. We take advantage of these two tests to verify the integrity of N both modulo p and modulo q (lines 17 and 20 of Alg. 11).

Remark 3. Note that we also made this version of the countermeasure infective, using the transformation method that we exposed in Sec. 3.2. As we said, any countermeasure can be transformed this way, for instance Alg. 13 in the Appendix B presents an infective variant of Aumüller *et al.*'s countermeasure.

Algorithm 11: CRT-RSA with our simplified Vigilant's countermeasure, under its infective avatar (new algorithm contributed in this paper)

Input : Message M, key (p, q, d_p, d_q, i_q)
Output: Signature $M^d \mod N$, or a random value in $\mathbb{Z}_N$

1 Choose a small random integer r.
2 $N = p \cdot q$
3 $p' = p \cdot r^2$
4 $i_{pr} = p^{-1} \mod r^2$
5 $M_p = M \mod p'$
6 $B_p = p \cdot i_{pr}$
7 $A_p = 1 - B_p \mod p'$
8 $M'_p = A_p \cdot M_p + B_p \cdot (1 + r) \mod p'$ // CRT insertion of verification value in M'_p
9 $q' = q \cdot r^2$
10 $i_{qr} = q^{-1} \mod r^2$
11 $M_q = M \mod q'$
12 $B_q = q \cdot i_{qr}$
13 $A_q = 1 - B_q \mod q'$
14 $M'_q = A_q \cdot M_q + B_q \cdot (1 + r) \mod q'$ // CRT insertion of verification value in M'_q
15 $S'_p = {M'_p}^{d_p \mod \varphi(p')} \mod p'$ // Intermediate signature in $\mathbb{Z}_{pr^2}$
16 $S_{pr} = 1 + d_p \cdot r$ // Checksum in $\mathbb{Z}_{r^2}$ for S'_p
17 $c_p = M'_p + N - M + 1 \mod p$
18 $S'_q = {M'_q}^{d_q \mod \varphi(q')} \mod q'$ // Intermediate signature in $\mathbb{Z}_{qr^2}$
19 $S_{qr} = 1 + d_q \cdot r$ // Checksum in $\mathbb{Z}_{r^2}$ for S'_q
20 $c_q = M'_q + N - M + 1 \mod q$
21 $S' = S'_q + q \cdot (i_q \cdot (S'_p - S'_q) \mod p')$ // Recombination in $\mathbb{Z}_{Nr^2}$
22 $S_r = S_{qr} + q \cdot (i_q \cdot (S_{pr} - S_{qr}) \mod p')$ // Recombination checksum in $\mathbb{Z}_{r^2}$
23 $c_S = S' - S_r + 1 \mod r^2$
24 **return** $S = S'^{c_p c_q c_S} \mod N$ // Retrieve result in $\mathbb{Z}_N$

6 Discussion: Limitations of our Formalism

Some CRT-RSA implementations that are proved secure in this article have been subject to published attacks. Such attacks actually exploit artifacts not captured by our modelization.

We mention below two of them, which are quite insightful about the difficulty to formally capture a security notion:

- one where the inputs are correlated (Vigilant, in [Vig08b, Slides 13–14]), which allows to make the redundant verifications ineffective;
- one where the effect of the fault is considered small enough to enumerate all of them, coupled with an inversion of an infective protection.

Of course, it is difficult to resist an omnipotent attacker. Despite our proof framework's ability to adapt to different attacks (by using different *attack success condition*), it still only captures the fault model we described in Def. 1 and 2, which may not always match reality. We nevertheless notice that the use of formal methods helps guarantee a baseline security level.

6.1 Correlated Inputs

We suppose that the inputs can be chosen by the attacker[7] and that the redundancy parameter r is known[8]. In this case, Vigilant has shown in his slides at CHES '08 that the countermeasure of Aumüller *et al.* [ABF+02] can be defeated as if unprotected by using for input m a multiple of r. Indeed, all the computations happening in the overrings $\mathbb{Z}_{pr}$ or $\mathbb{Z}_{qr}$ are then equal to zero modulo r, and thus alterations of the control flow graph remain undetected.

6.2 Low Entropy Infection

In [Wag04], Wagner attacks an infective protection, namely that called BOS [BOS03]. By assuming that the error is not uniformly distributed, but rather of low Hamming weight, he shows how to exploit the infected output by an exhaustive enumeration of plausible faults. Clearly, this is a drawback of infective protections. We can ignore it provided two hypotheses are formulated:

1. we assume that the mixing between the result and the fault-dependent quantity (denoted checks c_i, *e.g.*, in Alg. 5) is one-way, *i.e.*, the non-infected value cannot be recovered from the infected value and exhaustive guesses on the c_i, and/or
2. we assume that the attacker cannot accurately choose his fault model.

7 Conclusions and Perspectives

We studied the existing CRT-RSA algorithm countermeasures against fault-injection attacks, in particular the ones of Shamir's family. In so doing, we got a deeper understanding of their ins and outs. We obtained a few intermediate results: the absence of conceptual distinction between test-based and infective countermeasures, the fact that faults on the code (skipping instructions) can be captured by considering only faults on the data, and the fact that the many countermeasures that we studied (and their variations) were actually applying a common protection strategy but optimized it in different ways. These intermediate results allowed us to describe the design of a high-order countermeasure against our very generic fault model (comprised of randomizing, zeroing, and skipping faults). Our design allows to build a countermeasure resisting n faults for any n at a very reduced cost (it consists in adding $n-1$ comparisons on small numbers). We were also able to fix Shamir's countermeasure, and to drastically improve the one of Vigilant, going from 9

[7] Chosen plaintext is an example of threat that is not captured by our model.

[8] In practice, r can be chosen randomly for each execution, thus mitigating this attack.

verifications in the original countermeasure to only 3, removing computations made useless, and reducing its need for randomness, while at the same time making it infective instead of test-based.

Except for those which rely on the fact that the protected algorithm takes the form of a CRT computation, the ideas presented in the various countermeasures can be applied to any modular arithmetic computation. For instance, it could be done using the idea of Vigilant consisting in using the CRT to embed a known subring value in the manipulated numbers to serve as a checksum. That would be the most obvious perspective for future work, as it would allow a generic approach against fault attacks and even automatic insertion of the countermeasure.

A study of Giraud's family of countermeasures in more detail would be beneficial to the community as well.

Acknowledgment

We would like to thank Antoine Amarilli for his proofreading which greatly improved the editorial quality of our manuscript.

References

[ABF+02] Christian Aumüller, Peter Bier, Wieland Fischer, Peter Hofreiter, and Jean-Pierre Seifert. Fault Attacks on RSA with CRT: Concrete Results and Practical Countermeasures. In Burton S. Kaliski, Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 260–275. Springer, 2002.

[BDF+14] Gilles Barthe, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Mehdi Tibouchi, and Jean-Christophe Zapalowicz. Making RSA-PSS Provably Secure Against Non-Random Faults. *IACR Cryptology ePrint Archive*, 2014:252, 2014.

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the Importance of Checking Cryptographic Protocols for Faults. In *Proceedings of Eurocrypt'97*, volume 1233 of *LNCS*, pages 37–51. Springer, May 11-15 1997. Konstanz, Germany. DOI: 10.1007/3-540-69053-0_4.

[BNP07] Arnaud Boscher, Robert Naciri, and Emmanuel Prouff. CRT RSA Algorithm Protected Against Fault Attacks. In Damien Sauveron, Constantinos Markantonakis, Angelos Bilas, and Jean-Jacques Quisquater, editors, *WISTP*, volume 4462 of *Lecture Notes in Computer Science*, pages 229–243. Springer, 2007.

[BOS03] Johannes Blömer, Martin Otto, and Jean-Pierre Seifert. A new CRT-RSA algorithm secure against bellcore attacks. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM Conference on Computer and Communications Security*, pages 311–320. ACM, 2003.

[CCGV13] Maria Christofi, Boutheina Chetali, Louis Goubin, and David Vigilant. Formal verification of a CRT-RSA implementation against fault attacks. *Journal of Cryptographic Engineering*, 3(3):157–167, 2013.

[CGM+10] Jean-Sébastien Coron, Christophe Giraud, Nicolas Morin, Gilles Piret, and David Vigilant. Fault Attacks and Countermeasures on Vigilant's RSA-CRT Algorithm. In Luca Breveglieri, Marc Joye, Israel Koren, David Naccache, and Ingrid Verbauwhede, editors, *FDTC*, pages 89–96. IEEE Computer Society, 2010.

[CJ05] Mathieu Ciet and Marc Joye. Practical fault countermeasures for chinese remaindering based RSA. In *Fault Diagnosis and Tolerance in Cryptography*, 2005.

[CM09] Jean-Sébastien Coron and Avradip Mandal. PSS Is Secure against Random Fault Attacks. In *ASIACRYPT*, volume 5912 of *LNCS*, pages 653–666. Springer, December 6-10 2009. Tōkyō, Japan.

[DGRS09] Emmanuelle Dottax, Christophe Giraud, Matthieu Rivain, and Yannick Sierra. On Second-Order Fault Analysis Resistance for CRT-RSA Implementations. In Olivier Markowitch, Angelos Bilas, Jaap-Henk Hoepman, Chris J. Mitchell, and Jean-Jacques Quisquater, editors, *WISTP*, volume 5746 of *Lecture Notes in Computer Science*, pages 68–83. Springer, 2009.

[Gar65] Harvey L. Garner. Number Systems and Arithmetic. *Advances in Computers*, 6:131–194, 1965.

[Gir06] Christophe Giraud. An RSA Implementation Resistant to Fault Attacks and to Simple Power Analysis. *IEEE Trans. Computers*, 55(9):1116–1120, 2006.

[JPY01] Marc Joye, Pascal Paillier, and Sung-Ming Yen. Secure evaluation of modular functions, 2001.

[JT11] Marc Joye and Michael Tunstall. *Fault Analysis in Cryptography*. Springer LNCS, March 2011. `http://joye.site88.net/FAbook.html`. DOI: 10.1007/978-3-642-29656-7 ; ISBN 978-3-642-29655-0.

[KKHH11] Sung-Kyoung Kim, Tae Hyun Kim, Dong-Guk Han, and Seokhie Hong. An efficient CRT-RSA algorithm secure against power and fault attacks. *J. Syst. Softw.*, 84:1660–1669, October 2011.

[Koç94] Çetin Kaya Koç. High-Speed RSA Implementation, November 1994. Version 2, `ftp://ftp.rsasecurity.com/pub/pdfs/tr201.pdf`.

[LKW06] Sining Liu, Brian King, and Wei Wang. A CRT-RSA algorithm secure against hardware fault attacks. In *Second International Symposium on Dependable Autonomic and Secure Computing (DASC 2006), 29 September - 1 October 2006, Indianapolis, Indiana, USA*, pages 51–60. IEEE Computer Society, 2006.

[LRT14] Duc-Phong Le, Matthieu Rivain, and Chik How Tan. On double exponentiation for securing RSA against fault analysis. In Josh Benaloh, editor, *CT-RSA*, volume 8366 of *Lecture Notes in Computer Science*, pages 152–168. Springer, 2014.

[RG14a] Pablo Rauzy and Sylvain Guilley. A formal proof of countermeasures against fault injection attacks on CRT-RSA. *Journal of Cryptographic Engineering*, 4(3):173–185, 2014.

[RG14b] Pablo Rauzy and Sylvain Guilley. Formal Analysis of CRT-RSA Vigilant's Countermeasure Against the BellCoRe Attack. In *3rd ACM SIGPLAN Program Protection and Reverse Engineering Workshop (PPREW 2014)*, January 25 2014. San Diego, CA, USA. ISBN: 978-1-4503-2649-0.

[Riv09] Matthieu Rivain. Securing RSA against Fault Analysis by Double Addition Chain Exponentiation. Cryptology ePrint Archive, Report 2009/165, 2009. `http://eprint.iacr.org/2009/165/`.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[Sha99] Adi Shamir. Method and apparatus for protecting public key schemes from timing and fault attacks, November 1999. Patent Number 5,991,415; also presented at the rump session of EUROCRYPT '97.

[TW12] Mohammad Tehranipoor and Cliff Wang, editors. *Introduction to Hardware Security and Trust*. Springer, 2012. ISBN 978-1-4419-8079-3.

[Vig08a] David Vigilant. RSA with CRT: A New Cost-Effective Solution to Thwart Fault Attacks. In Elisabeth Oswald and Pankaj Rohatgi, editors, *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 130–145. Springer, 2008.

[Vig08b] David Vigilant. RSA with CRT: A New Cost-Effective Solution to Thwart Fault Attacks. In *CHES*, 2008. Slides presented at CHES [Vig08a] (personal communication).

[Wag04] David Wagner. Cryptanalysis of a provably secure CRT-RSA algorithm. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick Drew McDaniel, editors, *ACM Conference on Computer and Communications Security*, pages 92–97. ACM, 2004.

[YJ00] Sung-Ming Yen and Marc Joye. Checking Before Output May Not Be Enough Against Fault-Based Cryptanalysis. *IEEE Trans. Computers*, 49(9):967–970, 2000. DOI: 10.1109/12.869328.

A Recovering d and e from (p, q, d_p, d_q, i_q)

We prove here the following proposition:

Proposition 5. *It is possible to recover the private exponent d and the public exponent e from the 5-tuple (p, q, d_p, d_q, i_q) described in Sec. 2.2.*

Proof. Clearly, $p-1$ and $q-1$ are neither prime, nor coprimes (they have at least 2 as a common factor). Thus, proving Prop. 5 is not a trivial application of the Chinese Remainder Theorem. The proof we provide is elementary, but to our best knowledge, it has never been published before.

The numbers $p_1 = \frac{p-1}{\gcd(p-1,q-1)}$ and $q_1 = \frac{q-1}{\gcd(p-1,q-1)}$ are coprime, but there product is not equal to $\lambda(N)$. There is a factor $\gcd(p-1, q-1)$ missing, since $\lambda(N) = p_1 \cdot q_1 \cdot \gcd(p-1, q-1)$.

Now, $\gcd(p-1, q-1)$ is expected to be small. Thus, the following Alg. 12 can be applied efficiently. In this algorithm, the invariant is that p_2 and q_2, initially equal to p_1 and p_2, remain coprime. Moreover, they keep on increasing whereas r_2, initialized to $r_1 = \gcd(p-1, q-1)$, keeps on decreasing till 1.

Algorithm 12: Factorization of $\lambda(N)$ into two coprimes, multiples of p_1 and q_1 respectively.

Input : $p_1 = \frac{p-1}{\gcd(p-1,q-1)}$, $q_1 = \frac{q-1}{\gcd(p-1,q-1)}$ and $r_1 = \gcd(p-1, q-1)$
Output: (p_2, q_2), coprime, such as $p_2 \cdot q_2 = \lambda(N)$

$(p_2, q_2, r_2) \leftarrow (p_1, q_1, r_1)$
$g \leftarrow \gcd(p_2, r_2)$
while $g \neq 1$ **do**
  $p_2 \leftarrow p_2 \cdot g$
  $r_2 \leftarrow r_2/g$
  $g \leftarrow \gcd(p_2, r_2)$
end
$g \leftarrow \gcd(q_2, r_2)$
while $g \neq 1$ **do**
  $q_2 \leftarrow q_2 \cdot g$
  $r_2 \leftarrow r_2/g$
  $g \leftarrow \gcd(q_2, r_2)$
end
// p_2, q_2 and r_2 are now coprime
$q_2 \leftarrow q_2 \cdot r_2$
// $p_2 \leftarrow p_2 \cdot r_2$ would work equally
$(r_2 \leftarrow r_2/r_2 = 1)$
// For more pedagogy
return (p_2, q_2)

Let us denote p_2 and q_2 the two outputs of Alg. 12, we have:

- $d_{p_2} = d_p \mod p_2$, since $p_2|(p-1)$;
- $d_{q_2} = d_q \mod q_2$, since $q_2|(q-1)$;
- $i_{12} = {p_2}^{-1} \mod q_2$, since p_2 and q_2 are coprime.

We can apply Garner's formula to recover d:

$$d = d_{p_2} + p_2 \cdot ((i_{12} \cdot (d_{q_2} - d_{p_2})) \mod q_2) \ . \tag{1}$$

By Garner, we know that $0 \leq d < p_2 \cdot q_2 = \lambda(N)$, which is consistent with the remark made in the last sentence of Sec. 2.1.

Once we know the private exponent d, the public exponent e can be computed as the inverse of d modulo $\lambda(N)$. □

B Infective Aumüller CRT-RSA

The infective variant of Aumüller protection against CRT-RSA is detailed in Alg. 13.

Algorithm 13: CRT-RSA with Aumüller *et al.*'s countermeasure[6], under its infective avatar (new algorithm contributed in this paper)

Input : Message M, key (p, q, d_p, d_q, i_q)
Output: Signature $M^d \mod N$, or a random value

1 Choose a small random integer r.
2 $p' = p \cdot r$
3 $c_1 = p' + 1 \mod p$
4 $q' = q \cdot r$
5 $c_2 = q' + 1 \mod q$
6 $S'_p = M^{d_p \mod \varphi(p')} \mod p'$ // Intermediate signature in $\mathbb{Z}_{pr}$
7 $S'_q = M^{d_q \mod \varphi(q')} \mod q'$ // Intermediate signature in $\mathbb{Z}_{qr}$
8 $S_p = S'_p \mod p$ // Retrieve intermediate signature in $\mathbb{Z}_p$
9 $S_q = S'_q \mod q$ // Retrieve intermediate signature in $\mathbb{Z}_q$
10 $S = S_q + q \cdot (i_q \cdot (S_p - S_q) \mod p)$ // Recombination in $\mathbb{Z}_N$
11 $c_3 = S - S'_p + 1 \mod p$
12 $c_4 = S - S'_q + 1 \mod q$
13 $S_{pr} = S'_p \mod r$ // Checksum of S_p in $\mathbb{Z}_r$
14 $S_{qr} = S'_q \mod r$ // Checksum of S_q in $\mathbb{Z}_r$
15 $c_5 = {S_{pr}}^{d_q \mod \varphi(r)} - {S_{qr}}^{d_p \mod \varphi(r)} + 1 \mod r$
16 **return** $S^{c_1 c_2 c_3 c_4 c_5}$

Using Modular Extension to Provably Protect ECC Against Fault Attacks

Pablo Rauzy[2,1], Martin Moreau[1], Sylvain Guilley[1], and Zakaria Najm[1]

[1] Telecom ParisTech ; Institut Mines-Telecom ; CNRS LTCI
firstname.lastname@telecom-paristech.fr
[2] Inria — *firstname.lastname*@inria.fr

Abstract. Fault injection attacks are a real-world threat to cryptosystems, in particular asymmetric cryptography. In this paper, we focus on countermeasures which guarantee the integrity of the computation result, hence covering most existing and future faults attacks. Namely, we study the *modular extension* protection scheme in previously existing and newly contributed variants of the countermeasure on elliptic curve scalar multiplication (ECSM) algorithms. We find that an existing countermeasure is incorrect and we propose new "test-free" variant of the modular extension scheme that fixes it. We then formally prove the correctness and security of modular extension: specifically, the fault non-detection probability is inversely proportional to the security parameter. Finally, we implement an ECSM protected with test-free modular extension on an ARM Cortex-M4 microcontroller. A systematic fault injection campaign for several values of the security parameter confirms our theoretical prediction about the security of the obtained implementation, and provides figures for practical performance.

Keywords: fault injection attack, countermeasure, asymmetric cryptography, elliptic curve cryptography, modular extension.

1 Introduction

Properly used cryptography is a key building block for secure information exchange. Thus, implementation-level hacks must be considered seriously in addition to the threat of cyber-attacks. In particular, fault injection attacks target physical implementations of secured devices in order to induce exploitable errors.

Formal methods. In cryptology, formal methods aim at providing a mathematical / mechanical proof of security, which helps in building trust into proved cryptosystems. However, their use is still limited in the field of fault injection and side channel attacks as formal methods rely on models, and implementations are difficult to model properly.

Asymmetric cryptography. Asymmetric cryptography addresses different needs such as key exchange and digital signature. RSA, Diffie-Hellman, and ElGamal have been used for decades, and elliptic curve cryptography (ECC) algorithms

such as ECDSA [22] are more and more deployed. ECC pairing-based cryptography has recently been accelerated in practice and is thus becoming practical [37]. For example, the construction of "pairing-friendly" elliptic curves is an active subject [20]. Homomorphic encryption schemes are getting more practical and are progressively considered viable solutions for some real-world applications requiring strong privacy. All these algorithms use large numbers and take place in mathematical structures such as finite rings and fields. This property enables the use of formal methods but also facilitates attacks.

Fault Attacks. As put forward in the reference book on fault analysis in cryptography [25, Chp. 9], there are three main categories of fault attacks.
1) *Safe-error attacks* consist in testing whether an intermediate variable is dummy (usually introduced against simple power analysis [30]) or not, by faulting it and looking whether there is an effect on the final result.
2) *Cryptosystem parameter alterations* with the goal of weakening the algorithm in order to facilitate key extraction. For example in ECC, invalid-curve fault attacks consist in moving the computation to a weaker curve, enabling the attacker to use cryptanalysis attacks exploiting the faulty outputs.
3) Finally, the most serious attacks belong to the *differential fault analysis* (DFA) category. Often the attack path consists in comparing correct and faulted outputs, like in the well-known BellCoRe attack on CRT-RSA (RSA speeded up using the Chinese Remainder Theorem), or the sign-change fault attack on ECC.

The *BellCoRe attack* [9] on CRT-RSA introduced the concept of fault injection attacks. It is very powerful: faulting the computation even in a very random way yields almost certainly an exploitable result allowing to recover the secret primes of the RSA modulus $N = pq$. This attack is recalled in Sec. A for the sake of completeness.

The *sign-change attack* [8] on ECC consists in changing the sign of an intermediate elliptic curve point in the midst of an elliptic curve scalar multiplication (ECSM). The resulting faulted point is still on the curve so the fault is not detected by traditional point validation countermeasures. Such a fault can be achieved by for instance changing the sign in the double operation of the ECSM algorithm (line 3 of Alg. 1). If the fault injection occurs during the last iteration of the loop, then the final result $\widehat{Q} = [-2\sum_{i=1}^{n-1} k_i 2^{i-1}]P + k_0 P = -Q + 2k_0 P$, i.e., either $\widehat{Q} = -Q$ or $\widehat{Q} = -Q + 2P$ depending on k_0, which reveals the value of k_0 to the attacker. This process can be iterated to find the other bits of the scalar, and optimizations exist that trade-off between the number of necessary faulted results and the required exhaustive search.

Both RSA and ECC algorithms continue to be the target of **many** new fault injection attacks: see [3,31,5,6,15] just for some 2014 papers. Besides, this topic is emerging and other new fault attacks will appear sooner or later. Hence, the need for efficient and practical generic countermeasures against fault attacks is obvious. David Wagner from UC Berkeley concurs in [43]: "*It is a fascinating research problem to establish a principled foundation for security against fault attacks and to find schemes that can be proven secure within that framework.*"

Input : $P \in E$, $k = \sum_{i=0}^{n-1} k_i 2^i$ (n is the scalar size in bits, where $k_i \in \{0, 1\}$)
Output : $[k]P$

1 $Q \leftarrow \mathcal{O}$
2 **for** $i \leftarrow n - 1$ ***down to*** 0 **do**
3 $\quad Q \leftarrow 2Q$ ▷ ECDBL
4 $\quad$ **if** $k_i = 1$ **then** $Q \leftarrow Q + P$ ▷ ECADD
5 **return** Q

Algorithm 1: Double-and-add left-to-right scalar multiplication on elliptic curve E.

Countermeasures. Verifications compatible with mathematical structures can be applied either at computational or at algorithmic level.

Algorithmic protections have been proposed by Giraud [18] (and many others [10,32,28]) for CRT-RSA, which naturally transpose to ECC, as shown in [27]. These protections are implementation specific (e.g., depend on the chosen exponentiation algorithm) and are thus difficult to automate, requiring specialized engineering skills.

Computational protections have been pioneered by Shamir in [39] using *modular extension*, initially to protect CRT-RSA. The idea is to carry out the same computation in two different algebraic structures allowing to check the computation before disclosing its result. For example protecting a computation in $\mathbb{F}_p$ consists in carrying out the same computation in $\mathbb{Z}_{pr}$ and $\mathbb{F}_r$ ($\mathbb{Z}_{pr}$ is the direct product of $\mathbb{F}_p$ and $\mathbb{F}_r$), where r is a small number ($r \ll p$); the computation in $\mathbb{Z}_{pr}$ must match that of $\mathbb{F}_r$ when reduced modulo r, if not an error is returned, otherwise the result in $\mathbb{Z}_{pr}$ is reduced modulo p and returned. The principle of modular extension is sketched in Fig. 1. This method operates at low level (integer arithmetic), thereby enabling countermeasures (and optimizations) to be added on top of it. They are thus easily maintained, which explains why this method is quite popular. Indeed, there is a wealth of variants for CRT-RSA stemming from this idea [1,42,24,7,11,13], as well as a few proofs-of-concept transposing it to ECC [8,2,23]. Despite the nonexistence of literature, the same idea could apply to post-quantum code-based cryptography, pairing, and homomorphic computation for instance. Therefore, our paper focuses on computational countermeasures.

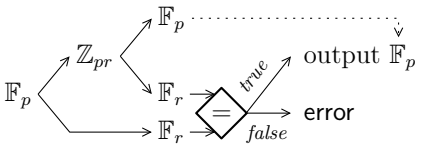


Fig. 1: Sketch of the principle of *modular extension*.

On the one hand, the variety of CRT-RSA countermeasures shows that fault attacks are a threat that is taken seriously by both the academic and the industrial communities. On the other hand, it bears witness to the artisanal way these countermeasures were put together. Indeed, the absence of formal security claims and of proofs added to the necessity of writing implementations by hand results in many weaknesses in existing countermeasures and thus in many attempts to create better ones.

Contributions. We study the *modular extension* protection scheme in existing countermeasures on elliptic curve scalar multiplication (ECSM) algorithms, namely BOS [8] and BV [2].

We show that BOS is incorrect (Sec. 3.1), i.e., that in some specific cases that we strictly characterize, it does not return the expected result (even in the absence of fault injections), which may induce a security issue. The flaw in BOS is reminiscent to that provoked artificially by means of injecting *points with low order neighbours* and *bitflip faults* in [16].

We show that BV is correct (Sec. 3.2), but that it is weaker and more vulnerable to fault injections, i.e., that in some specific cases that we strictly characterize, faults are not detected. As it happens, these specific cases are exactly the same as the ones where BOS returns an incorrect result.

We introduce the notion of *test-free algorithms* (Sec. 4) as a solution to BOS incorrectness, and then use it to propose a test-free variant of BOS (TF-BOS). We prove that TF-BOS is correct.

We then formally study the security of the test-free variant of modular extension (Sec. 5) and show that the fault non-detection probability is inversely proportional to the security parameter.

Finally, we implement TF-BOS on an ARM Cortex-M4 microcontroller and perform a systematic fault injection campaign for several values of the security parameter (Sec. 6), which confirms the security of the countermeasure, and provides figures for its practical performance.

2 ECSM on the Projective Plane

Definition 1 (Elliptic curve over a finite field). *An elliptic curve is a plane curve over a finite field $\mathbb{F}_p$, which is denoted $E(\mathbb{F}_p)$ (or simply E when the base field is implicit). It is composed of a specified point, called "point at infinity" and denoted by $\mathcal{O}$, and of the points (x, y) satisfying an equation of the form $y^2 = x^3 + ax + b$ (known as Weierstrass equation), where the discriminant $\Delta = -16(4a^3 + 27b^2)$ is nonzero. Alongside with elliptic curve group operations, this set of points form an additive group, where $\mathcal{O}$ is the identity element.*

The points of the curve can be represented in a coordinate system over $\mathbb{F}_p$, the most natural representation being affine. However, in such system, operations on the curves are complicated due to divisions. Thus, we focus on a kind of representation known as *projective*[1], where a third coordinate Z is added, so as to avoid divisions. The equation of the curve thus becomes: $Y^2Z = X^3 + aXZ^2 + bZ^3$, where $X = xZ$ and $Y = yZ$. By convention, $\mathcal{O}$ is represented by $(X : Y : 0)$ in the projective plane. Remark that the Z coordinate is redundant: we can get rid of it by a so-called projective-to-affine transformation, which maps $(X : Y : Z)$ to $(x = X/Z, y = Y/Z)$.

[1] Other projective coordinate systems exist, such as that of Jacobi, but for the sake of simplicity and without loss of generality, we focus on the projective system.

Definition 2 (Curve order). *The order $\#E$ of an elliptic curve E is the number of points on the curve.*

Definition 3 (Point order). *The order $ord(P)$ of a point P on a elliptic curve E is the smallest non-null integer k such that $[k]P = \mathcal{O}$. The maximum value of $ord(P)$ is $\#E$.*

Definition 4 (Generator). *Let $P \in E$. The point P is called a generator of E if $E = \{[k]P, 0 \leq k < \#E\}$, or equivalently if $ord(P) = \#E$.*

Remark 1. The coordinates $(X : Y : Z)$ normally belong to a finite field $\mathbb{F}_p$, but for the purpose of the modular extension countermeasure, we extend the notion of elliptic curve to rings (such as $\mathbb{Z}_{pr}$). For this reason we use the metavariable n (and $\mathbb{Z}_n$) in some algorithms to represent an integer rather than p or r (and $\mathbb{F}_p$ or $\mathbb{F}_r$) which we use to represent prime numbers.

Computer algebra tools (e.g., MAGMA or SAGE) refuse to handle elliptic curves on $\mathbb{F}_n$ when n is composite. However, in projective coordinates, computations do not involve divisions, hence ECSM can be computed. Projective versions of point doubling, point addition, and scalar multiplication are detailed in Sec. B.1.

3 State-of-the-Art on ECSM Protection Against Fault Attacks with Modular Extension

Definition 5 (Correct algorithm). *An algorithm is said correct if it returns the right result when no faults have been injected.*

3.1 BOS

In [8], Blömer, Otto, and Seifert propose a countermeasure based on the modular extension idea of Shamir for CRT-RSA [39]. It is presented in Alg. 2.

Input : $P \in E(\mathbb{F}_p)$, $k \in \{1, \ldots, ord(P) - 1\}$
Output : $Q = [k]P \in E(\mathbb{F}_p)$

1 Choose a small prime r, a curve $E(\mathbb{F}_r)$, and a point P_r on that curve.
2 Determine the combined curve $E(\mathbb{Z}_{pr})$ and point P_{pr} using the CRT.[2]
3 $(X_{pr} : Y_{pr} : Z_{pr}) = \text{ECSM}(P_{pr}, k, pr)$
4 $(X_r : Y_r : Z_r) = \text{ECSM}(P_r, k, r)$
5 **if** $(X_{pr} \bmod r : Y_{pr} \bmod r : Z_{pr} \bmod r) = (X_r : Y_r : Z_r)$ **then**
6 | **return** $(X_{pr} \bmod p : Y_{pr} \bmod p : Z_{pr} \bmod p)$
7 **else**
8 | **return** error

Algorithm 2: ECSM protected with BOS countermeasure $\text{BOS}(P, k, p)$.

[2] See Sec. D.1.

An issue with BOS, which is not visible here as we purposedly presented a division-free version of the ECSM algorithm, is that their paper does not address the problem of divisions in $\mathbb{Z}_{pr}$. We will show that it is actually possible to circumvent this problem if necessary in Sec. 5.1. However, there is also a correction and security issue with BOS.

Proposition 1. *BOS is incorrect.*

Proof. There are tests in the ECDBL (Alg. 6) and ECADD (Alg. 7) algorithms called from the ECSM algorithm (Alg. 8). The latter is called twice by BOS: once to compute the ECSM on the combined curve (line 3 of Alg. 2), and once to compute the ECSM on the small curve (line 4 of Alg. 2). The conditions of the tests in ECDBL and ECADD depend on the inputs (point and scalar) of the ECSM, and thus are not satisfied at the same time in the small and in the combined computations. Indeed, the order of the point P_r on the small curve is much smaller than k, so the small computation may come across $\mathcal{O}$ and may satisfy some of the tests, while this is not going to happen on the combined curve even if taken modulo r it should. As a result, the operations carried out in $E(\mathbb{Z}_{pr})$ are not the same as in $E(\mathbb{F}_r)$, thus the comparison on line 5 of Alg. 2 may fail. In such cases, BOS will return error while the result in $E(\mathbb{F}_p)$ is actually good. □

This behavior can be a *serious security issue* as it reveals information about the inputs. We will see in Sec. 4 that the leaked information can be very precise about the scalar. A numerical example where BOS outputs an incorrect result is given in Sec. D.2.

In 2010 Joye patented [23] essentially the same countermeasure except it uses $\mathbb{F}_{r^2}$ and $\mathbb{Z}_{pr^2}$ instead of $\mathbb{F}_r$ and $\mathbb{Z}_{pr}$, which does not address the raised issues.

3.2 BV

In [2], Baek and Vasyltsov propose a countermeasure based on modular extension and point verification. The problem of divisions is explicitly evaded by carrying out computations in Jacobian coordinates. For the sake of simplicity, we use the BV protection scheme with projective coordinates. It is presented in Alg. 3.

The particularity of BV is that instead of computing a sibling ECSM on a smaller curve $E(\mathbb{F}_r)$ to compare with its redundant counterpart over $E(\mathbb{Z}_{pr})$, it only checks whether the point obtained by reducing the result $E(\mathbb{Z}_{pr})$ modulo r is on the $E(\mathbb{F}_r)$ curve (i.e., whether it satisfies the curve equations modulo r).

Proposition 2. *BV is correct.*

Proof. The incorrectness of BOS comes from the fact that the small computations hits $\mathcal{O}$ and triggers some of the tests in ECDBL and ECADD, while in the combined computation the conditions of the tests are not satisfied even if the same thing happens modulo r.

[3] See Sec. D.3 which details the curve and Jacobian equation originally used by BV.

Input : $P \in E(\mathbb{F}_p)$, $k \in \{1, \ldots, ord(P) - 1\}$
Output : $Q = [k]P \in E(\mathbb{F}_p)$

1 Choose a small random integer r.
2 Compute the combined curve $E'(\mathbb{Z}_{pr})$.[3]
3 $(X_{pr} : Y_{pr} : Z_{pr}) = \text{ECSM}(P, k, pr)$
4 **if** $Y_{pr}^2 Z_{pr} = X_{pr}^3 + aX_{pr}Z_{pr}^2 + bZ_{pr}^3 \mod r$ **then**
5 | **return** $(X_{pr} \bmod p : Y_{pr} \bmod p : Z_{pr} \bmod p)$
6 **else**
7 | **return** error

Algorithm 3: ECSM protected with BV countermeasure BV(P, k, p).

Instead of the broken comparison, BV verifies that the point is on the curve modulo r. In the problematic case when the point modulo r is $\mathcal{O}$, we have that $Z_{pr} \equiv 0 \bmod r$, so the equation on line 4 of Alg. 3 is violated only if $X_{pr} \not\equiv 0 \bmod r$. We will see in the proof of Prop. 3 in the next section that this is never the case. □

Note that the correctness of BV comes with a drawback: indeed, faults may go undetected if they happen before $\mathcal{O}$ is reached in the computation modulo r as the intermediate point quickly tends to $(0 : 0 : 0)$ and stay there until the end. This claim will be underpinned in the next section.

Besides, we note that BV is more flexible with respect to the choice of the r parameter (line 1 of Alg. 3) than BOS (line 1 of Alg. 2). However, as will be underlined in Sec. 5.3, this is a *security weakness* of BV: the parameter r should preferentially be chosen to be a prime. Notice that this choice would nonetheless invalidate BV as a countermeasure against side-channel attacks (as claimed in the original paper), as prime numbers of given bitwidth are not uniformly distributed.

3.3 Conclusion

We can now compare BOS and BV.

- BOS is incorrect and because of that may leak information on its inputs, even in the absence of fault attacks. Besides, BOS also has a problem with divisions in the $\mathbb{Z}_{pr}$ ring;
- BV is correct, however it employs a point verification technique that is of course specific to elliptic curve computations and is thus less generic (in the sense that the countermeasure is not trivially portable to any other modular computation).

An obvious question is now: is it possible to get the best of both worlds, i.e., a generic countermeasure that relies on the classical modular extension scheme (Fig. 1), and is correct? The answer is yes, as we will see in the next section which introduces "test-free algorithms" to fix BOS correctness issue.

4 Test-Free Algorithms

The correctness issue of BOS comes from the fact that the conditions of the tests in the ECDBL (Alg. 6) and ECADD (Alg. 7) algorithms are not satisfied at the same time in the small and the combined computations. Because of that, these computations do not perform the same sequence of operations, invalidating the modular extension invariant.

4.1 Test-Free ECSM

A simple fix is getting rid of the conditional tests. This simplification engenders *partial domain correctness* as exposed in [16]. We detail the *test-free* variants of Alg. 6, 7, and 8 in Alg. 9, 10, and 11, which can be found in Sec. B.2. Actually, "test-free" refers to the absence of point comparison (line 1 of Alg. 6, and lines 1, 2, 3, and 4 of Alg. 7). The test depending on the scalar value (i.e., at line 4 of Alg. 11) is still present as it will always be satisfied at the same time in both computations (in $\mathbb{Z}_{pr}$ and $\mathbb{F}_r$), since the same scalar is used.

The ECSM $[k]P$ has no tests which condition is satisfied if k is a TF-good scalar (see Prop. 3 for a rigorous proof).

Definition 6 (TF-good scalar). *Let $P \in E(\mathbb{Z}_n)$. Let $k > 0$. The scalar k is said to be TF-good with regard to P and $E(\mathbb{Z}_n)$ if and only if:*

1. $ord(P) \not| \lfloor k/2^i \rfloor$, *for* $\lceil \log_2 k \rceil - 1 \geq i \geq 1$, *and for* $i = 0$ *when* $k_0 = 1$,
2. $ord(P) \not| \lfloor k/2^i \rfloor - 1$, *for* $\lceil \log_2 k \rceil - 1 \geq i \geq 0$ *when* $k_i = 1$,
3. $ord(P) \not| \lfloor k/2^i \rfloor - 2$, *for* $\lceil \log_2 k \rceil - 1 \geq i \geq 0$ *when* $k_i = 1$.

Remark that a scalar k such that $0 < k < ord(P)$ is always TF-good.

This definition of TF-good scalar is relative to the left-to-right ECSM algorithm, but our results are portable to the other variants as well. Interestingly, the same definition would apply to the left-to-right add-always ECSM algorithm [12, §3.1]. However, the definition would differ for other variants; for instance, the case of the right-to-left ECSM algorithm is detailed in Sec. C.2, and generally more ECSM algorithms are treated in Sec. C.

The relevance of TF-good concept is given in Prop. 3.

Proposition 3 (Partial domain correctness of TF-ECSM (Alg. 11 at page 29)). *Let $P = (X_P : Y_P : Z_P) \in E(\mathbb{Z}_n)$, and $k > 0$. We have:*

1. *if k is TF-good with regard to P and $E(\mathbb{Z}_n)$ then TF-ECSM(P, k, n) = ECSM(P, k, n);*
2. *Otherwise, TF-ECSM(P, k, n) is $\mathcal{O}$, specifically: it has the form $(0 : Y : 0)$, where $Y = 0$ except if $k_0 = 1$ and $ord(P) \mid k$.*

Proof. We start with the first point. We want to prove that when k is TF-good, then the test-free versions of the algorithms return the same results as the original version. We will prove this by showing that it is equivalent to say "k is TF-good" and "during the execution none of the conditions of the tests in the original algorithms are satisfied", and thus the tests can be removed safely.

The only conditional test in ECDBL (line 1 of Alg. 6) checks whether the point $[\lfloor k/2^{i+1} \rfloor]P$ given as argument is $\mathcal{O}$. This condition will never be met when k is TF-good by point 1 of Def. 6.

The same reasoning applies to the first conditional test in ECADD (line 1 of Alg. 7), but with point 2 of Def. 6. Indeed, the value of Q in the ECADD is $[2\lfloor k/2^{i+1} \rfloor]P = [\lfloor k/2^i \rfloor - 1]P$ because $k_i = 1$.

The second conditional test in ECADD (line 2 of Alg. 7) checks whether the point given as argument to ECSM is $\mathcal{O}$, in which case $ord(P) = 1$, and all three conditions in Def. 6 are violated.

The third conditional test in ECADD (line 3 of Alg. 7) checks if $Q = -P$, that is if $P + Q = \mathcal{O}$. Let us suppose that is the case. It would mean that after the ECADD (if this test was removed), we would be in the situation where the point given as argument to ECDBL is $\mathcal{O}$, which we already have shown to be impossible by point 1 of Def. 6.

The fourth and last conditional test in ECADD (line 4 of Alg. 7) checks if $Q = P$, that is if $P - Q = \mathcal{O}$. Let us suppose that is the case. Before entering ECADD, we know that $k_i = 1$ and that $Q = [\lfloor k/2^i \rfloor - 1]P$. Now, since $P - Q = \mathcal{O}$ it means that $ord(P) \mid \lfloor k/2^i \rfloor - 2$, which contradicts point 3 of Def. 6.

This proves point 1. Let's now prove point 2 by studying what happens when the condition of one of the removed tests would be satisfied.

We call "step u" the ECSM loop iteration where $i = u$. We note $Q_u = (X_u : Y_u : Z_u)$ the value of the intermediate point Q at the end of step u, the final result being $(X_0 : Y_0 : Z_0)$.

We assume that condition 1 in Def. 6 is violated. If it is only violated for $i = 0$ and $k_0 = 1$, then $[k]P = \mathcal{O}$. In this case, the last operation is ECADD on $[k-1]P = -P$ and P. So, as can be seen in Alg. 7, $B = 0$, hence the result takes the form $(0 : Y_0 : 0)$.

If the condition 1 in Def. 6 is violated for $i > 0$, then $[\lfloor k/2^i \rfloor]P = \mathcal{O}$. If this value has been obtained by an ECDBL, $Z_i = 0 \implies X_i = 0$ (refer to Alg. 6). If this value has been obtained by an ECADD, then according to the previous argument ($B = 0$ in Alg. 7), we also have $Z_i = X_i = 0$. Now, as $i > 0$, the computation continues with at least one ECDBL, which results in $(0 : 0 : 0)$. Hence, all forecoming computations result in $(0 : 0 : 0)$.

Let us now assume that the condition 2 in Def. 6 is violated. If it is violated for $i = 0$, then $k-1$ is even, and in the last ECDBL, we have $Z = 0 \implies X = 0$. In the last ECADD, we have $B = C = 0$, hence the result is $(0 : 0 : 0)$. The same reasoning can be done for $i > 0$.

Finally, let us assume that the condition 3 in Def. 6 is violated. For $i = 0$, this means that $ord(P) \mid k - 2$ when $k_0 = 1$, hence $[k-1]P = P$. So, at the last ECDBL, we have $A = B = 0 \implies C = 0$, and the result is equal to $(0 : 0 : 0)$. The same holds for $i > 1$, since $(0 : 0 : 0)$ is a fixed point.

Therefore, each time k is not TF-good, the result is $(0 : Y_0 : 0)$, where Y_0 is null, except if k is even and is equal to $ord(P)$. □

4.2 Simplifying BV

When computing $[k]P$ on the curve $E(\mathbb{F}_p)$, the scalar k is usually chosen such that $0 < k < ord(P)$. Consequently, according to Def. 6, k is always TF-good with regard to P. Thus, by Prop. 3 we can use the test-free variants of the doubling and addition algorithms for BV without loss of generality. We can now give a rigorous proof of the correctness of BV (Prop. 2).

Proof. When k is TF-good with regard to $P \bmod r$ on $E'(\mathbb{Z}_{pr}) \bmod r$, we know that BV gives the correct output.

When k is TF-bad[4] with regard to $P \bmod r$ on $E'(\mathbb{Z}_{pr}) \bmod r$, we know from Prop. 3 that the result is of the form $(0 : Y : 0)$, which satisfies line 4 of Alg. 3.

When $k = ord(P)$ and $k_0 = 0$, k is TF-good but one may wonder what happens since the result is $\mathcal{O}$. In this case, reusing the notation from Prop. 3, we have $Z_0 = 0$, i.e., $8Z_1^3Y_1^3 = 0$. We know that $Z_1 \neq 0$ because Q_1 is not $\mathcal{O}$, which means that $Y_1 = 0$, which implies that $X_0 = 0$, and thus that the equation on line 4 of Alg. 3 is satisfied as expected. □

The test-free variant of the ECSM algorithm thus allows for a correct simplification of BV[5], but more interestingly, it allows to fix the correctness problem of BOS.

4.3 Fixing BOS

We now propose to fix BOS countermeasure by using the test-free variant of the elliptic curve algorithms. We call this new countermeasure TF-BOS.

Input : $P \in E(\mathbb{F}_p)$, $k \in \{1, \ldots, ord(P) - 1\}$
Output : $Q = [k]P \in E(\mathbb{F}_p)$

1 Choose a small prime r, a curve $E(\mathbb{F}_r)$, and a point P_r on that curve.
2 Determine the combined curve $E(\mathbb{Z}_{pr})$ and point P_{pr} using the CRT.
3 $(X_{pr} : Y_{pr} : Z_{pr}) = \text{TF-ECSM}(P, k, pr)$
4 $(X_r : Y_r : Z_r) = \text{TF-ECSM}(P_r, k, r)$
5 **if** $(X_{pr} \bmod r : Y_{pr} \bmod r : Z_{pr} \bmod r) = (X_r : Y_r : Z_r)$ **then**
6 | **return** $(X_{pr} \bmod p : Y_{pr} \bmod p : Z_{pr} \bmod p)$
7 **else**
8 | **return** error

Algorithm 4: TF-ECSM with modular extension protection TF-BOS(P, k, p).

Proposition 4. *TF-BOS is correct.*

[4] We say that a scalar is TF-bad if it is not TF-good.

[5] Recall that a test, e.g. $P = Q$ (where P and Q are in projective representation), requires four products, since $P = Q$ is equivalent to $(Z_P = Z_Q = 0) \vee ((X_P Z_Q = X_Q Z_P) \wedge (Y_P Z_Q = Y_Q Z_P))$, which has a non-negligible cost.

Proof. Whether k is TF-bad or not, the result in $E(\mathbb{Z}_{pr})$ reduced modulo r and the result in $E(\mathbb{F}_r)$ will always match during the computation, and in particular when it ends. Thus, the modular extension invariant verification done on line 5 of Alg. 4 will always be satisfied in the absence of faults. □

Intuitively, what Prop. 4 says is that TF-BOS is the correct way to implement modular extension *in general*. However, the correctness of TF-BOS comes with the same drawback as BV's correctness: it reduces the fault detection probability, albeit in a quantifiable and negligible manner in practice.

We remark the interesting duality between BOS and TF-BOS: in the same cases where BOS is incorrect, TF-BOS is blind to fault injections. Indeed, when TF-BOS is used with a TF-bad scalar, the computation in $E(\mathbb{Z}_{pr})$ taken modulo r will, at some iteration of the ECSM, be equal to $\mathcal{O}$. Now, as shown in the proof of the second point of Prop. 3, once a point is equal to $\mathcal{O}$ it remains so, and worse, its coordinates become $(0:0:0)$ with a single additional iteration (owing to the fact that tests, especially tests of equality to $\mathcal{O}$, are absent in the *test-free* version of the ECSM). Now, if any coordinate is faulted to nonzero value, the other coordinates might stay at zero, which contaminates the result to $(0:0:0)$ again after one TF-ECADD or TF-ECDBL. A quantitative analysis of impact of faults under TF-bad scalar is given in Sec. 6.3, in particular in Prop. 7.

Proposition 5 (Probability of TF-bad scalars). *The probability of a scalar k to be TF-bad with respect to a point $P \in E(\mathbb{F}_r)$ is $O\left(\frac{1}{ord(P)}\right)$.*

Proof. Let $n = \lceil \log_2 k \rceil$ the size of k and $m = \lceil \log_2 ord(P) \rceil$ the size of $ord(P)$.

Using the same notation as in Def. 6, for all $i < m$ all of the three conditions defining TF-good scalars are met. For each $i \geq m$, there are $(2^{i+1} - 2^i) = 2^i$ numbers of size i bits, out of which a fraction of approximately $\frac{2^i}{ord(P)}$ violates one of the TF-good conditions (say the 1st condition).

For k to be TF-bad it suffices that it exists an i for which one of the TF-good conditions is violated, so the probability of k being TF-bad is:

$$\begin{aligned}\mathbb{P}_{\text{TF-bad}_P}(k) &\approx 1 - \left(\prod_{m \leq i \leq n} 1 - \frac{\frac{2^i}{ord(P)}}{2^i} \right) = 1 - \left(1 - \frac{1}{ord(P)}\right)^{n-m} \\ &= O\left(\frac{1}{ord(P)}\right).\end{aligned}$$

□

In the context of the modular extension countermeasure against fault injection attacks (recall Fig. 1), TF-bad scalars are more likely to occur in the small field $\mathbb{F}_r$ than in the large field $\mathbb{F}_p$ or the large ring $\mathbb{Z}_{pr}$. Indeed, an elliptic curve over $\mathbb{F}_r$ has about r points, with $r \ll p$, but the scalar k has about size p. Indeed, for instance in ECDH and ECDSA, k is chosen uniformly in $\{1, \ldots, ord(P) - 1\}$, where $ord(P) \approx p$. We have this lemma:

Lemma 1. *Let $E(\mathbb{F}_p)$ an elliptic curve over $\mathbb{F}_p$ given by equation $y^2 = x^3+ax+b \mod p$ (recall Def. 1), and $P = (x_P, y_P)$ a point on this curve. Then the set $E(\mathbb{F}_r)$ of pairs $(x_r, y_r) \in \mathbb{F}_r$ satisfying $y_r^2 = x_r^3 + a_r x_r + b_r \mod r$, with $a_r = a \mod r$ and $b_r = (y_P^2 - x_P^3 - ax_P) \mod r$. is an elliptic curve over $\mathbb{F}_r$, and $P_r = (P \mod r) = (x_P \mod r, y_P \mod r)$ belongs to it.*

Proof. Clearly, by Def. 1, $E(\mathbb{F}_r)$ is an elliptic curve (provided the discriminant $-16(4a_r^3 + 27b_r^2)$ is nonzero). Besides, as $P \in E(\mathbb{F}_p)$, there exists an integer λ such that $y_P^2 = x_P^3 + ax_P + b + \lambda p$. Therefore, modulo r, we have $(y_P \mod r)^2 = (x_P \mod r)^3 + (a \mod r)(x_P \mod r) + (b + \lambda p \mod r)$, hence $P_r = (x_P \mod r, y_P \mod r) \in E(\mathbb{F}_r)$, if $b_r = b + \lambda p \mod r = (y_P^2 - x_P^3 - ax_P) \mod r$. □

We compute some examples based on curve P-192 based on $\mathbb{F}_p$ (p being a 192 bit prime), which will be our running example in Sec. 6. The parameters of this curve are recalled in Sec. 6.1. We assume k is on 192 bits, and we choose small prime numbers $r \ll p$ as in Sec. 6.2. Using Lem. 1 we derive a curve over $\mathbb{F}_r$ and a point from P-192. We then compute, thanks to Prop. 5, the probability of a scalar k to be TF-bad. Results obtained from SAGE are given in Tab. 1.

Table 1: Illustration of Prop. 5 for some small r values.

r	$ord(P_r)$	$\mathbb{P}_{\text{TF-bad}_{P_r}}(k)$
251	267	$5.0 \cdot 10^{-1}$
1021	509	$3.0 \cdot 10^{-1}$
2039	2105	$8.2 \cdot 10^{-2}$
4093	4041	$4.4 \cdot 10^{-2}$
65521	65531	$2.7 \cdot 10^{-3}$
4294967291	2147439270	$7.5 \cdot 10^{-8}$
18446744073709551557	18446744077549890349	$6.9 \cdot 10^{-18}$

It can be seen in Tab. 1 that for $r = 1021$ and $r = 4294967291$, the order of P_r is not maximal. Actually, for those two curves, P_r is not a generator of $E(\mathbb{F}_r)$, and its order is half the number of points of the curve.

This justifies the recommendation made in BOS [8, Sec. 4] to build $E(\mathbb{F}_r)$ as a curve with large order and P_r as a generator (see Appendix D.1).

4.4 Using Edwards Curves

Edwards curves [14] have been studied in the context of cryptography by Bernstein and Lange [4]. On Edwards curves, the addition law is *complete*: addition formulas work for all pairs of input points. In particular, there is no troublesome point at infinity. Therefore, there is no such ECADD and ECDBL for Edwards curves, but one *test-free* formula for addition (the input points being equal or not). This formula is given in [4, Sec. 4, page 9] for projective coordinates. Therefore, we underline that Edwards curves are especially well suited for the modulus extension countermeasure, since any avatar (such as BOS or BV) is correct and there is no possibility of TF-bad scalars.

4.5 Conclusion

All the algorithms proposed in this section are correct and offer the same security level. In particular, we note that when k is TF-good with regard to the small computation, all three countermeasures are correct and secure[6]. Moreover, it follows trivially from Def. 6 that it is possible to statically determine if a given scalar is TF-bad with regard to a point and its curve. In addition, it is important to note that in practice with $r \approx 2^{32}$, the problems with TF-bad scalars become anecdotal (see Prop. 5).

Edwards curves are definitely a nice option to implement fault detection using modular extension, since all scalars are TF-good (there is no tests in Edwards curves). However, as real-world industrial applications are still based on Weierstrass curves, we continue the paper by taking them as examples. Hence, in the sequel, we assume curves have TF-bad scalars.

5 Formal Security Study of Modular Extension

5.1 Inversions in Direct Products

We will start by addressing the issue of divisions in $\mathbb{Z}_{pr}$. It is actually possible to circumvent this problem in the modular extension setting. Indeed, divisions can be optimized, as expressed in the following proposition.

Proposition 6 (Divisions optimization). *To get the inverse of z in $\mathbb{F}_p$ while computing in $\mathbb{Z}_{pr}$, one has:*
- $z = 0 \bmod r \implies (z^{p-2} \bmod pr) \equiv z^{-1} \mod p$,
- *otherwise* $(z^{-1} \bmod pr) \equiv z^{-1} \mod p$.

Proof. If $z = 0 \mod r$, then z is not invertible in $\mathbb{Z}_{pr}$. However, z^{p-2} exists in $\mathbb{Z}_{pr}$, and $(z^{p-2} \bmod pr) \bmod p = z^{p-2} \bmod p = z^{-1} \bmod p$. Notice that, as p is statically known, a precomputed efficient addition chain can be used.

Otherwise, when $z \neq 0 \mod r$, we have in $\mathbb{Z}_{pr}$ that $z^{-1} = z^{\varphi(pr)-1} = z^{pr-p-r} \mod pr$. Now, $(z^{-1} \bmod pr) \bmod p = z^{-1} \bmod p$ if and only if: $\varphi(p)$ divides $(pr - p - r) - (-1)$. But $(pr - p - r) - (-1) = (p-1)(r-1)$, which is indeed a multiple of $\varphi(p) = p - 1$. □

Notice that in Proposition 6, we assume $0 \leq z < p$. Indeed, in practical ECC computations, the scalar is chosen such that it is smaller than the generator point (base point) order. Therefore, no "division by 0" is supposed to show up in $\mathbb{F}_p$. Typically, as divisions occur only in projective to affine conversions, we have that the Z coordinate (whose value matches z in Proposition 6) is non-zero because the final point is not at infinity.

[6] In the sense that the fault detection probability is maximal for the modular extension method.

Remark 2. Golić and Tymen introduce in [19] a masking countermeasure of the advanced encryption standard (AES), called the "Embedded Multiplicative Masking", which also requires to embed a finite field into a larger ring. In this context, the over-structure is a *polynomial extension* of some extension of $\mathbb{F}_2$, but the idea is similar to *modular extension*. In particular, the authors notice in section 5.1 of their paper [19] that inversion in the base field can be obtained in the over-ring as an exponentiation to the base field order minus two.

But the inversion procedure we give in Proposition 6 is novel, in that we allow an optimization if the number is inversible in the over-ring. This requires a test, which we can do safely without disclosing information in the context of fault attacks detection. Nonetheless, such optimization would be insecure in the context of the "Embedded Multiplicative Masking" countermeasure, since this would leak information about the value of the mask. This is a first-order flaw which would undermine the security of the "Embedded Multiplicative Masking" protection against side-channel attacks.

The section E discusses the complexity of inversions as in Prop. 6. An upper-bound for the expected overhead is $(10 \times (1 - \frac{1}{r}) + 384 \times \frac{1}{r})/10 \approx 1 + 10^{-8}$ when r is a 32 bit number, which is negligible in practice.

5.2 Security Analysis

Definition 7 (Fault model). *We consider an attacker who can fault data by randomizing or zeroing any intermediate variable, and fault code by skipping any number of consecutive instructions.*

Definition 8 (Attack order). *We call order of the attack the number of faults (in the sense of Def. 7) injected during the target execution.*

In the rest of this section, we focus mainly on the resistance to first-order attacks on data. Indeed, Rauzy and Guilley have shown in [38] that 1. it is possible to adapt the modular extension protection scheme to resist attack of order D for any D by chaining D repetitions of the final check in a way that forces each repetition of the modular extension invariant verification to be faulted independently, and 2. faults on the code can be formally captured (simulated) by faults on intermediate variables.

Definition 9 (Secure algorithm). *An algorithm is said secure if it is correct as per Def. 5 and if it either returns the right result or an* error *constant when faults have been injected, with an overwhelming probability.*

Theorem 1 (Security of the test-free modular extension scheme). *Test-free algorithms protected using the modular extension technique, such as TF-BOS, are secure as per Def. 9. In particular, the probability of non-detection is inversely proportional to the security parameter r.*

Proof. Faulted results are polynomials of faults. The result of an asymmetric cryptography computation can be written as a function of a subset of the intermediate variables, plus some inputs if the intermediate variables do not suffice to finish the computation. We are interested in the expression of the result as a function of the intermediate variables which are the target of a transient or permanent fault injection. We give the formal name $\widehat{x}$ to any faulted variable x. For convenience, we denote them by $\widehat{x_i}$, $1 \leq i \leq n$, where $n \geq 1$ is the the number of injected faults. The result consists in additions, subtractions, and multiplications of those formal variables (and inputs). Such expression is a multivariate polynomial. If the inputs are fixed, then the polynomial has only n formal variables. We call it $P(\widehat{x_1}, \ldots, \widehat{x_n})$. For now, let us assume that $n = 1$, i.e., that we face a single fault. Then P is a monovariate polynomial. Its degree d is the multiplicative depth of $\widehat{x_1}$ in the result.

A fault is not detected if and only if $P(\widehat{x_1}) = P(x_1) \mod r$, whereas $P(\widehat{x_1}) \neq P(x_1) \mod p$. Notice that the latter condition is superfluous insofar since if it is negated then the effect of the fault does not alter the result in $\mathbb{F}_p$.

Non-detection probability is inversely proportional to r. As the faulted variable $\widehat{x_1}$ can take any value in $\mathbb{Z}_{pr}$, the non-detection probability $\mathbb{P}_{\text{n.d.}}$ is given by:

$$\begin{aligned}\mathbb{P}_{\text{n.d.}} &= \frac{1}{pr-1} \cdot \sum_{\widehat{x_1} \in \mathbb{Z}_{pr} \setminus \{x_1\}} \delta_{P(\widehat{x_1}) \,=\, P(x_1) \mod r} \\ &= \frac{1}{pr-1} \cdot \Big(-1 + p \sum_{\widehat{x_1}=0}^{r-1} \delta_{P(\widehat{x_1}) \,=\, P(x_1) \mod r} \Big). \end{aligned} \tag{1}$$

Here, $\delta_{\text{condition}}$ is equal to 1 (resp. 0) if the condition is true (resp. false).

Let $\widehat{x_1} \in \mathbb{Z}_r$, if $P(\widehat{x_1}) = P(x_1) \mod r$, then $\widehat{x_1}$ is a root of the polynomial $\Delta P(\widehat{x_1}) = P(\widehat{x_1}) - P(x_1)$ in $\mathbb{Z}_r$. We denote by $\#\text{roots}(\Delta P)$ the number of roots of ΔP over $\mathbb{Z}_r$. Thus (1) computes $(p \times \#\text{roots}(\Delta P) - 1)/(pr-1) \approx \#\text{roots}(\Delta P)/r$.

Study of the proportionality constant. A priori, bounds on this value are broad since $\#\text{roots}(\Delta P)$ can be as high as the degree d of ΔP in $\mathbb{Z}_r$, i.e., $\min(d, r-1)$. However, in practice, ΔP looks like a random polynomial over the finite field $\mathbb{Z}_r$, for several reasons:

- inputs are random numbers in most cryptographic algorithms, such as probabilistic signature schemes,
- the coefficients of ΔP in $\mathbb{Z}_r$ are randomized due to the reduction modulo r.

In such case, the number of roots is very small, despite the possibility of d being large. See for instance [34] for a proof that the number of roots tends to 1 as $r \to \infty$. Interestingly, random polynomials are still friable (i.e., they are clearly not irreducible) in average, but most factors of degree greater than one happen not to have roots in $\mathbb{Z}_r$. Thus, we have $\mathbb{P}_{\text{n.d.}} \gtrsim \frac{1}{r}$, meaning that $\mathbb{P}_{\text{n.d.}} \geq \frac{1}{r}$ but is close to $\frac{1}{r}$. A more detailed study of the theoretical upper bound on the number of roots is available in Sec. F.

The same law applies to multiple faults. In the case of multiple faults ($n > 1$), then the probability of non-detection generalizes to:

$$\begin{aligned}
\mathbb{P}_{\text{n.d.}} &= \tfrac{1}{(pr-1)^n} \cdot \sum_{\widehat{x_1},\ldots,\widehat{x_n} \in \mathbb{Z}_{pr}\setminus\{x_1\}\times\ldots\times\mathbb{Z}_{pr}\setminus\{x_n\}} \delta_{P(\widehat{x_1},\ldots,\widehat{x_n})=P(x_1,\ldots,x_n) \bmod r} \\
&= \tfrac{1}{(pr-1)^n} \cdot \sum_{\widehat{x_2},\ldots,\widehat{x_n} \in \prod_{i=2}^{n} \mathbb{Z}_{pr}\setminus\{x_i\}} \left[\sum_{\widehat{x_1} \in \mathbb{Z}_{pr}\setminus\{x_1\}} \delta_{P(\widehat{x_1},\ldots,\widehat{x_n})=P(x_1,\ldots,x_n) \bmod r} \right] \\
&= \tfrac{1}{(pr-1)^n} \cdot \sum_{\widehat{x_2},\ldots,\widehat{x_n} \in \prod_{i=2}^{n} \mathbb{Z}_{pr}\setminus\{x_i\}} [p \times \#\text{roots}(\Delta P) - 1] \\
&= \tfrac{1}{(pr-1)^n} \cdot (pr-1)^{n-1} [p \times \#\text{roots}(\Delta P) - 1] \\
&= \frac{p \times \#\text{roots}(\Delta P) - 1}{pr-1}.
\end{aligned}$$

Therefore, the probability not to detect a fault when $n > 1$ is identical to that for $n = 1$. Thus, we also have $\mathbb{P}_{\text{n.d.}} \approx \frac{1}{r}$ in the case of multiple faults of the intermediate variables[7]. □

Examples can be found in Sec. G that illustrate the security property: indeed, $\mathbb{P}_{\text{n.d.}}$ is inversely proportional to r, with a proportionality constant which depends on the specific algorithm. The purpose of Sec. 6 is to show that the product $\mathbb{P}_{\text{n.d.}} \times r$ is constant in practice. Moreover, we explicit this constant for ECSM computations.

5.3 Modular Extension the Right Way

Here we recap how to correctly implement the modular extension countermeasure in order to achieve maximum security.

Vocabulary. We call *nominal* computation the original unprotected computation over $\mathbb{F}_p$. We call *small* computation the computation performed over a smaller field $\mathbb{F}_r$. We call *combined* computation the same computation lifted into the direct product $\mathbb{Z}_{pr}$ of $\mathbb{F}_p$ and $\mathbb{F}_r$.

Test-free algorithms. Protecting a computation using a modular extension based countermeasure means that this computation will be run twice for comparison; and thus the operation performed in the combined and in the small computation must be the equivalent. When there are tests depending on the data in $\mathbb{F}_p$ in the nominal computation, the conditions of these tests may not be satisfied at the same time in the small and in the combined computations, which would make the modular extension invariant check fail while no faults have been injected (as we have seen in BOS countermeasure for example). This problem can be circumvented by using a test-free version of the computation. Of course this may

[7] Note that this study does not take correlated faults into account.

cause the test-free computation to only be correct for a part of the domain of the nominal computation. Hopefully, the part of the domain for which the computation becomes incorrect can be statically determined and is negligible in practice for ECSM (and does not exists for CRT-RSA which nominal computation is naturally test-free, idem for ECC on Edwards curves).

Conditions on the security parameter. The size r of the mathematical structure underlying the small computation is the security parameter of a modular extension countermeasure. Several conditions have to be met to obtain maximum security and the best performance:

- r must be co-prime with p, but as p is a larger prime this should never be a problem;
- r must be prime itself, in order to avoid the maximum possible division problems in the combined computation (performance), and for $\mathbb{F}_r$ to be a field, thus maximizing the chances of $ord(P)$ to be big enough in $E(\mathbb{F}_r)$ (security);
- r should be large enough for the non-detection probability to be sufficiently low, but at the same time should remain small enough to keep the overhead of the countermeasure reasonable (we have seen that for CRT-RSA and ECSM, a value of r on 32 bits is a good option);
- r may be static, but as pointed out in [2], it helps against against side-channel analyses if it is randomly selected at runtime;
- r should not be public information contrary to what is said in [8], as it would give an attacker the opportunity to forge input values which breaks the countermeasure (e.g., a point P such that $ord(P \bmod r)$ is very low in the case of ECSM, or a message which is a multiple of r in CRT-RSA);

These recommendations can be understood as "choose r so that $P \bmod r$ is of maximal order on $E(\mathbb{F}_r)$", which similar to what is suggested in the original BOS paper [8] (see also the same comment made after the presentation of Tab. 1).

6 Practical Case Study with TF-BOS

In order to practically validate our theoretical results, we have implemented TF-BOS (Alg. 4) on an ARM Cortex-M4 microcontroller (specifically an STM32). For the sake of simplicity, we decided to use $E(\mathbb{Z}_{pr}) \bmod r$ as $E(\mathbb{F}_r)$ and $P \bmod r$ as P_r. Hence, the security results may be slightly negatively impacted.

6.1 Setup

The code is written in C and uses the GMP library[8]. We used the P-192 elliptic curve from NIST [41, D.1.2.1]. This curve is the less secure amongst that proposed by NIST, but is chosen because our ARM chip is rather low-end and we

[8] We used the `mini-gmp` implementation for easy portability onto the ARM microcontroller.

nonetheless need a reasonable speed. Besides, we opted for a curve standardized by NIST because they are still widely used in the industry. Parameter values are listed below:

Field characteristic	p	= 0xffffffffffffffffffffffffffffffffeffffffffffffff
Curve equation	a	= 0xffffffffffffffffffffffffffffffffeffffffffffffffc
coefficients	b	= 0x64210519e59c80e70fa7e9ab72243049feb8deecc146b9b1
Point coordinates	x_P	= 0x188da80eb03090f67cbf20eb43a18800f4ff0afd82ff1012
	y_P	= 0x07192b95ffc8da78631011ed6b24cdd573f977a11e794811
Point order	$ord(P)$	= 0xffffffffffffffffffffffff99def836146bc9b1b4d22831

Fig. 2 shows an architectural overview of the electromagnetic fault injection (EMFI) analysis platform we used for our experiments. EMFI has recently emerged as an efficient non-invasive fault attack, which is able to perturb a circuit through its package. The platform includes a signal generator able to generate pulses of 1.5 ns width amplified by a broadband class A amplifier (400 MHz, 300 Watt max), and an electromagnetic (EM) probe. An oscilloscope and a data timing generator are also present, so that we can precisely (with 1 ps precision) control the delay before the injection. All experiments have been performed at a settled spatial location of the EM probe relative to the ARM microcontroller: a fixed position and a fixed angular orientation. A boundary-scan (also known as JTAG) probe has been used to dump internal registers and memory contents after injection (for attack analysis purpose only).

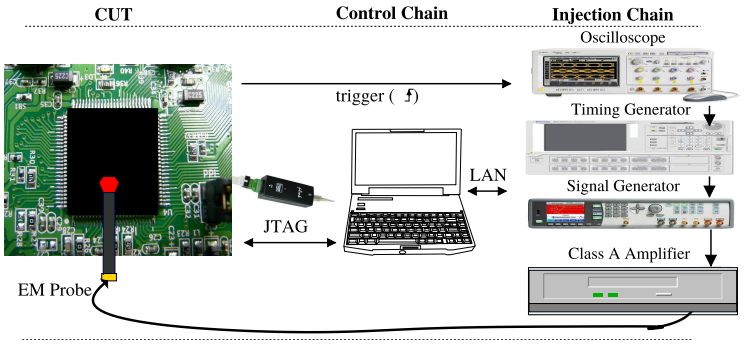


Fig. 2: EMFI platform.

We manually explored the effect of different width and power of the EM pulse, and chose values which maximize the faulting success rate. Then, we manually tuned the delay before the injection happens in order to maximize the probability of obtaining an exploitable fault for each value of r.

6.2 Method

In order to assess our theoretical results, we performed multiple attack campaigns with different values for r. The practical results allowed us to verify our theoretical predictions, i.e., that the probability of non-detection $\mathbb{P}_{\text{n.d.}}$ is inversely proportional to r (see Sec. 5.2). At the same time we were able to measure the

cost of the countermeasure and confirm that the size of r is a security parameter that trades off speed for security.

- The value $r = 1$ basically means that there is no countermeasure, since $\mathbb{F}_r = \mathbb{F}_1 = \{0\}$. It helps verify that the platform is indeed injecting faults effectively, i.e., that most of the fault injection attempts are successful.
- The small values of r (on 8 to 16 bits) aim at verifying that the probability of detection / non-detection follow our theoretical prediction.
- The values of r on 32 and 64 bits represent realistic values for an operational protection.

Each value of r is chosen to be the largest prime number of its size. That is, if n is the size of r in bits, then r is the largest prime number such that $r < 2^n$.

6.3 Security Results

Tab. 2 shows the security assessment of the TF-BOS countermeasure. For each value of r (lines of the table) we ran and injected random faults in approximately[9] 1000 ECSM $[k]P$ using a random 192-bit k. In total, the execution of the tests we present took approximately 6 hours of computation. The results of our attack campaign are depicted in the last four columns.

- Correct results for which there is no error detection are simply fault injections without effect (*true negatives*).
- Correct results for which an error is detected are *false positives*, and should be minimized. Those false positive alarms are annoyances, as they warn despite no secret is at risk security-wise.
- The incorrect results for which an error is detected (*true positives*) should appear with probability $O(1 - \frac{1}{r})$.
- The incorrect results for which there is no error detection are *false negatives*, and should really be minimized: otherwise, the countermeasure is bypassed without notice and sensitive information may leak.

Once renormalized to remove the true negatives, the last column of Tab. 2 (false negatives) represents the non-detection probability $\mathbb{P}_{\text{n.d.}}$. The relationship between r and $\mathbb{P}_{\text{n.d.}}$ is plotted in Fig. 3. The experimental results are the dots with error bars (representing plus/minus one sigma), and match the theoretical curve in blue color. The asymptotical equivalent, namely $r \mapsto 96/r$, is superimposed in red color, and is a valid approximation for $r \gtrsim 2000$, which is reasonable since practical values of r are $\approx 2^{32}$.

Proposition 7 ($\mathbb{P}_{\text{n.d.}}$ proportionality factor for TF-BOS). *In the case of TF-BOS on curve P-192, the proportionality constant is* ≈ 96.

Lemma 2. *During TF-ECSM (Alg. 11), if the coordinates of the intermediate point becomes multiples of r, they stay so until the end.*

[9] A bit less in practice: a few attempts were lost due to communication errors between our computer and the JTAG probe's `gdb-server`.

Table 2: TF-BOS security assessment results.

r value	r size	Positives (%)		Negatives (%)	
	(bit)	true	false	true	false
1	1	0.00	0.00	2.74	97.3
251	8	63.7	0.00	2.56	33.8
1021	10	89.1	0.00	2.96	7.95
2039	11	98.8	0.00	0.00	1.18
4093	12	97.6	0.00	1.91	0.48
65521	16	97.8	0.00	2.21	0.00
4294967291	32	97.2	0.00	2.81	0.00
18446744073709551557	64	99.8	0.00	0.21	0.00

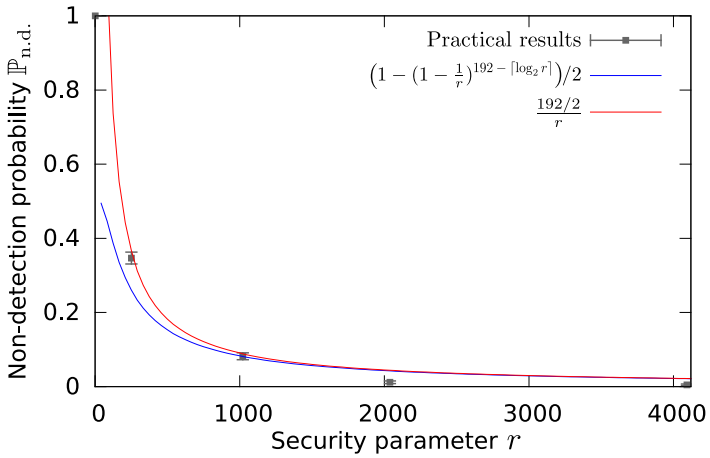


Fig. 3: Relationship between $\mathbb{P}_{\text{n.d.}}$ and r.

Proof. By Prop. 3, we know that when a coordinate becomes 0 it stays so until the end, for instance when k is TF-bad with regard to P_r on the small curve. In the combined computation on the curve $E(\mathbb{Z}_{pr})$, this translates to coordinates being null modulo r when k is TF-bad with regard to P on the small curve $E(\mathbb{F}_r)$. □

Proof. Using Lem. 2, we can now prove Prop. 7.

The first thing to note is that despite the fact that we are checking the validity of three coordinates modulo r, the $\mathbb{P}_{\text{n.d.}}$ is not $O(\frac{1}{r^3})$. That is because the three coordinates are extremely interdependent, fault-wise (as Lem. 2 shows, for instance): if one coordinate is faulted, then it is very likely that all coordinates are faulted. Actually it should be sufficient to perform the modular extension invariant check only on one of the coordinates, but we chose to still check all of them as it is virtually cost-free.

If a fault occurs before step i where k satisfies one of the TF-bad conditions, then the fault is not detected. Indeed in these cases, both the result of the ECSM on $E(\mathbb{Z}_{pr})$ modulo r and the result on $E(\mathbb{F}_r)$ are equal to $(0:0:0)$.

This happens with probability $\frac{1}{2}(1-(1-\frac{1}{ord(P)})^{192-\lceil \log_2 ord(P)\rceil})$, i.e., the probability of having a TF-bad scalar (see Prop. 5), with the $\frac{1}{2}$ factor to accounts for the faulting to happen before step i, where it is likely to be absorbed by a subsequent product with a multiple of r.

If we approximate the order of the point P on the $E(\mathbb{F}_r)$ curve by r, we have $\frac{1}{2}(1-(1-\frac{1}{r})^{192-\lceil \log_2 r\rceil}) = \frac{(192-\lceil \log_2 r\rceil)/2}{r} + O(\frac{1}{r})$. In practice, we can safely remove the "$-\lceil \log_2 r\rceil$" part as $\log_2 r$ will be of negligible value, especially once divided by 2. Thus we have $\mathbb{P}_{\text{n.d.}} \approx \frac{192/2}{r}$. □

As Tab. 2 and Fig. 3 show, practical values of r are sufficiently large for the latter equality to be true, and thus for the security to be highly efficient.

6.4 Performance Results

The table presented in Tab. 3 shows the cost of the modular extension countermeasure in terms of speed[10]. For each value of r (lines of the table) we list the execution time of the ECSM computation over $\mathbb{Z}_{pr}$, of the one over $\mathbb{F}_r$, of the test (comprising the extraction modulo r from the result of the computation over $\mathbb{Z}_{pr}$ and its comparison with the result of the computation over $\mathbb{F}_r$), and eventually the overhead of the countermeasure.

In the unprotected implementation, the ECSM computation over $\mathbb{F}_p$ took 683 ms (which naturally corresponds to the 683 ms over $\mathbb{Z}_{pr}$ when $r = 1$ as shown in Tab. 3, except that there is no need for the 24 ms needed by the computation over $\mathbb{F}_r$ which is mathematically trivial, but not optimized by `gcc`). We can see that when r is on 32 bits, the alignment with `int` makes `mini-gmp` faster, resulting in the protected algorithm running for 1004 ms, incurring a factor of only about $\times 1.47$ in the run time compared to the unprotected algorithm.

[10] Note that we compiled the code with `gcc -O0` option.

Table 3: TF-BOS performance results.

r value	r size (bit)	time (ms)			overhead
		$\mathbb{Z}_{pr}$	$\mathbb{F}_r$	test	
1	1	683	24	$\ll 1$	$\times 1.04$
251	8	883	91	$\ll 1$	$\times 1.43$
1021	10	899	100	$\ll 1$	$\times 1.46$
2039	11	902	197	$\ll 1$	$\times 1.61$
4093	12	903	197	$\ll 1$	$\times 1.61$
65521	16	883	189	$\ll 1$	$\times 1.56$
4294967291	32	832	172	$\ll 1$	$\times 1.47$
18446744073709551557	64	996	246	$\ll 1$	$\times 1.82$

This is a particularly good performance result. Indeed, in the context of digital signature, for instance ECDSA [22], an alternative to the verification by modular extension is the mere verification of the signature. However, the verification in ECDSA incurs an overhead of about $\times 4.5$ (measured with `openssl speed ecdsa` for P-192), indeed, in ECDSA, the signature verification is much more complex than the signature generation. Moreover, the curve P-192 that we use is among the smallest standardized curves, and the performance factor is directly tied to the increase in the size of the ring in which the computations are performed: when $\mathbb{F}_p$ grows, the countermeasure gets cheaper as $\frac{\log_2(pr)}{\log_2(p)}$ will be smaller.

7 Conclusion and Perspectives

In this paper, we have studied how to efficiently protect elliptic curve scalar multiplications (ECSM), against fault injection attacks. We have focused on countermeasures which guarantee the integrity of the computation result, hence covering most existing and future faults attacks.

Specifically, we have reviewed the state of the art of the *modular extension* protection scheme in existing countermeasures for ECSM algorithms, namely BOS [8] and BV [2]. We have shown that BOS is incorrect, while BV is correct but weaker against fault injection attacks.

We have introduced the notion of *test-free algorithms* as a simplification of BV and as a solution to fix the incorrectness of BOS. We call TF-BOS our contributed variant of the BOS modular extension based countermeasures. While TF-BOS fixes the correctness issue of BOS, it also inherits from one of the weakness of BV. We then proposed a characterization of the scalar argument of the ECSM with regard to the elliptic curve point argument as TF-good/TF-bad. Interestingly, it is in the same condition (when the scalar is TF-bad) that BOS returns an incorrect result and that BV and TF-BOS cannot detect fault injections.

We have formally studied the security of our proposed TF-BOS countermeasure, and proven that the fault non-detection probability is inversely proportional to the security parameter.

Finally, we implemented TF-BOS on an ARM Cortex-M4 microcontroller and performed a systematic fault injection campaign for several values of the security parameter, which confirmed the security of the countermeasure, and provided figures for its practical performance.

To our best knowledge, this is the first ECSM implementation to be provably protected against fault injection attacks. We used it to show that the cost of the TF-BOS countermeasure is extremely reasonable: **with a 32-bit value for the security parameter, the code is less than 1.5 times slower**. Our fault injection campaign revealed that it is also very efficient: using the same 32-bit security parameter, **100% of the fault injections were detected**.

A notable conclusion of our study is that with regard to protection against fault injection attacks, Edwards curve are the best choice for ECC, as the completeness of their addition formula avoids the existence of TF-bad scalars.

Our main perspective is that the test-free variant of the modular extension protection scheme is *generic* (e.g., it corresponds to all CRT-RSA countermeasures based on Shamir's idea). A natural extension would be its application to pairing. In this case, some computations must be carried out in *field extensions*, typically with embedding degree $m = 12$.

Finally, the security parameter r can be chosen randomly at execution time. As already pointed out in [2], this can make a natural protection against side-channel analyses. The formalization of this nice side-effect of the modular extension protection scheme would be welcomed.

References

1. C. Aumüller, P. Bier, W. Fischer, P. Hofreiter, and J.-P. Seifert. Fault Attacks on RSA with CRT: Concrete Results and Practical Countermeasures. In Kaliski et al. [26], pages 260–275.
2. Y.-J. Baek and I. Vasyltsov. How to Prevent DPA and Fault Attack in a Unified Way for ECC Scalar Multiplication - Ring Extension Method. In E. Dawson and D. Wong, editors, *Information Security Practice and Experience*, volume 4464 of *Lecture Notes in Computer Science*, pages 225–237. Springer Berlin Heidelberg, 2007.
3. G. Barthe, F. Dupressoir, P. Fouque, B. Grégoire, and J. Zapalowicz. Synthesis of Fault Attacks on Cryptographic Implementations. In G. Ahn, M. Yung, and N. Li, editors, *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pages 1016–1027. ACM, 2014.
4. D. J. Bernstein and T. Lange. Faster addition and doubling on elliptic curves. In K. Kurosawa, editor, *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings*, volume 4833 of *Lecture Notes in Computer Science*, pages 29–50. Springer, 2007.

5. J. Blömer, R. Gomes Da Silva, P. Gunther, J. Krämer, and J.-P. Seifert. A Practical Second-Order Fault Attack against a Real-World Pairing Implementation. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2014 Workshop on*, pages 123–136, Sept 2014. Busan, Korea.
6. J. Blömer, P. Günther, and G. Liske. Tampering Attacks in Pairing-Based Cryptography. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2014 Workshop on*, pages 1–7, Sept 2014. Busan, Korea.
7. J. Blömer, M. Otto, and J.-P. Seifert. A new CRT-RSA algorithm secure against bellcore attacks. In S. Jajodia, V. Atluri, and T. Jaeger, editors, *ACM Conference on Computer and Communications Security*, pages 311–320. ACM, 2003.
8. J. Blömer, M. Otto, and J.-P. Seifert. Sign Change Fault Attacks on Elliptic Curve Cryptosystems. In L. Breveglieri, I. Koren, D. Naccache, and J.-P. Seifert, editors, *Fault Diagnosis and Tolerance in Cryptography*, volume 4236 of *Lecture Notes in Computer Science*, pages 36–52. Springer Berlin Heidelberg, 2006.
9. D. Boneh, R. A. DeMillo, and R. J. Lipton. On the Importance of Checking Cryptographic Protocols for Faults. In *Proceedings of Eurocrypt'97*, volume 1233 of *LNCS*, pages 37–51. Springer, May 11-15 1997. Konstanz, Germany. DOI: 10.1007/3-540-69053-0_4.
10. A. Boscher, R. Naciri, and E. Prouff. CRT RSA Algorithm Protected Against Fault Attacks. In D. Sauveron, C. Markantonakis, A. Bilas, and J.-J. Quisquater, editors, *WISTP*, volume 4462 of *Lecture Notes in Computer Science*, pages 229–243. Springer, 2007.
11. M. Ciet and M. Joye. Practical fault countermeasures for chinese remaindering based RSA. In *Fault Diagnosis and Tolerance in Cryptography*, pages 124–131, Friday September 2nd 2005. Edinburgh, Scotland.
12. J.-S. Coron. Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In Ç. K. Koç and C. Paar, editors, *CHES*, volume 1717 of *LNCS*, pages 292–302. Springer, 1999.
13. E. Dottax, C. Giraud, M. Rivain, and Y. Sierra. On Second-Order Fault Analysis Resistance for CRT-RSA Implementations. In O. Markowitch, A. Bilas, J.-H. Hoepman, C. J. Mitchell, and J.-J. Quisquater, editors, *WISTP*, volume 5746 of *Lecture Notes in Computer Science*, pages 68–83. Springer, 2009.
14. Edwards, Harold M. A normal form for elliptic curves. *Bulletin of the American Mathematical Society*, 44:393–422, 9 April 2007. DOI: 10.1090/s0273-0979-07-01153-6, ISSN 0002-9904.
15. N. El Mrabet, J. J. Fournier, L. Goubin, and R. Lashermes. A survey of fault attacks in pairing based cryptography. *Cryptography and Communications*, pages 1–21, 2014.
16. J. Fan, B. Gierlichs, and F. Vercauteren. To Infinity and Beyond: Combined Attack on ECC Using Points of Low Order. In B. Preneel and T. Takagi, editors, *CHES*, volume 6917 of *LNCS*, pages 143–159. Springer, 2011.
17. H. L. Garner. Number Systems and Arithmetic. *Advances in Computers*, 6:131–194, 1965.
18. C. Giraud. An RSA Implementation Resistant to Fault Attacks and to Simple Power Analysis. *IEEE Trans. Computers*, 55(9):1116–1120, 2006.
19. J. D. Golić and C. Tymen. Multiplicative masking and power analysis of AES. In Kaliski et al. [26], pages 198–212.
20. A. Guillevic and D. Vergnaud. Genus 2 Hyperelliptic Curve Families with Explicit Jacobian Order Evaluation and Pairing-Friendly Constructions. In M. Abdalla and T. Lange, editors, *Pairing-Based Cryptography — Pairing 2012*, volume 7708

of *Lecture Notes in Computer Science*, pages 234–253. Springer Berlin Heidelberg, 2013.
21. D. Hankerson, A. J. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.
22. D. Johnson, A. Menezes, and S. Vanstone. The Elliptic Curve Digital Signature Algorithm (ECDSA). *International Journal of Information Security*, 1(1):36–63, 2001.
23. M. Joye. Fault-resistant calculations on elliptic curves, Sept. 15 2010. EP Patent App. EP20,100,155,001 ; `http://www.google.com/patents/EP2228716A1?cl=en`.
24. M. Joye, P. Paillier, and S.-M. Yen. Secure evaluation of modular functions. In R. Hwang and C. Wu, editors, *International Workshop on Cryptology and Network Security*, pages 227–229, September, 26-28 2001. `http://joye.site88.net/papers/JPY01dfa.pdf`, Taipei, Taiwan.
25. M. Joye and M. Tunstall. *Fault Analysis in Cryptography*. Springer LNCS, March 2011. `http://joye.site88.net/FAbook.html`. DOI: 10.1007/978-3-642-29656-7 ; ISBN 978-3-642-29655-0.
26. B. S. Kaliski, Jr., Ç. K. Koç, and C. Paar, editors. *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*. Springer, 2003.
27. D. Karaklajic, J. Fan, J. Schmidt, and I. Verbauwhede. Low-cost fault detection method for ECC using montgomery powering ladder. In *Design, Automation and Test in Europe, DATE 2011, Grenoble, France, March 14-18, 2011*, pages 1016–1021. IEEE, 2011.
28. S.-K. Kim, T. H. Kim, D.-G. Han, and S. Hong. An efficient CRT-RSA algorithm secure against power and fault attacks. *J. Syst. Softw.*, 84:1660–1669, October 2011.
29. Ç. K. Koç, T. Acar, and B. S. Kaliski, Jr. Analyzing and comparing montgomery multiplication algorithms. *Micro, IEEE*, 16(3):26–33, Jun 1996.
30. P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
31. R. Lashermes, M. Paindavoine, N. El Mrabet, J. J. Fournier, and L. Goubin. Practical Validation of Several Fault Attacks against the Miller Algorithm. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2014 Workshop on*, pages 115–122, Sept 2014. Busan, Korea.
32. D.-P. Le, M. Rivain, and C. H. Tan. On double exponentiation for securing RSA against fault analysis. In J. Benaloh, editor, *CT-RSA*, volume 8366 of *Lecture Notes in Computer Science*, pages 152–168. Springer, 2014.
33. A. Lenstra, H. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.
34. V. Leont'ev. Roots of random polynomials over a finite field. *Mathematical Notes*, 80(1-2):300–304, 2006.
35. M. McLoone, C. McIvor, and J. V. McCanny. Coarsely integrated operand scanning (CIOS) architecture for high-speed Montgomery modular multiplication. In O. Diessel and J. Williams, editors, *Proceedings of the 2004 IEEE International Conference on Field-Programmable Technology, Brisbane, Australia, December 6-8, 2004*, pages 185–191. IEEE, 2004.
36. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, October 1996.

37. M. Naehrig, R. Niederhagen, and P. Schwabe. New software speed records for cryptographic pairings. In M. Abdalla and P. S. Barreto, editors, *Progress in Cryptology – LATINCRYPT 2010*, volume 6212 of *Lecture Notes in Computer Science*, pages 109–123. Springer-Verlag Berlin Heidelberg, 2010. Updated version: `http://cryptojedi.org/papers/#dclxvi`.
38. P. Rauzy and S. Guilley. Countermeasures Against High-Order Fault-Injection Attacks on CRT-RSA. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2014 Workshop on*, pages 68–82, Sept 2014. Busan, Korea.
39. A. Shamir. Method and apparatus for protecting public key schemes from timing and fault attacks, November 1999. US Patent Number 5,991,415; also presented at the rump session of EUROCRYPT '97 (May 11–15, 1997, Konstanz, Germany).
40. U.S. Department of Commerce, National Institute of Standards and Technology. Recommended Elliptic Curves For Federal Government Use, July 1999. `http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf`.
41. U.S. Department of Commerce, National Institute of Standards and Technology. FIPS PUB 186-4, FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION: Digital Signature Standard (DSS), July 2013. `https://oag.ca.gov/sites/all/files/agweb/pdfs/erds1/fips_pub_07_2013.pdf`.
42. D. Vigilant. RSA with CRT: A New Cost-Effective Solution to Thwart Fault Attacks. In E. Oswald and P. Rohatgi, editors, *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 130–145. Springer, 2008.
43. D. Wagner. Cryptanalysis of a provably secure CRT-RSA algorithm. In V. Atluri, B. Pfitzmann, and P. D. McDaniel, editors, *ACM Conference on Computer and Communications Security*, pages 92–97. ACM, 2004.

A Unprotected CRT-RSA and the BellCoRe Attack

The *BellCoRe attack* [9] on CRT-RSA (RSA optimized using the Chinese Remainder Theorem) introduced the concept of fault injection attacks. It is very powerful: faulting the computation even in a very random way yields almost certainly an exploitable result allowing to recover the secret primes of the RSA modulus $N = pq$. Indeed, in the CRT-RSA algorithm, which is described in Alg. 5, most of the time is spent in the exponentiation algorithm. If the intermediate variable S_p (resp. S_q) is returned faulted as $\widehat{S_p}$ (resp. $\widehat{S_q}$), then the attacker gets an erroneous signature $\widehat{S}$, and is able to recover q (resp. p) as $\gcd(N, S - \widehat{S})$. For any integer x, $\gcd(N, x)$ can only be either 1, p, q, or N. In Alg. 5, if S_p is faulted (i.e., replaced by $\widehat{S_p} \neq S_p$), then $S - \widehat{S} = q \cdot ((i_q \cdot (S_p - S_q) \bmod p) - (i_q \cdot (\widehat{S_p} - S_q) \bmod p))$, and thus $\gcd(N, S - \widehat{S}) = q$. If S_q is faulted (i.e., replaced by $\widehat{S_q} \neq S_q$), then $S - \widehat{S} \equiv (S_q - \widehat{S_q}) - (q \bmod p) \cdot i_q \cdot (S_q - \widehat{S_q}) \equiv 0 \mod p$ because $(q \bmod p) \cdot i_q \equiv 1 \mod p$, and thus $S - \widehat{S}$ is a multiple of p. Additionally, the difference $(S - \widehat{S})$ is not a multiple of q. So, $\gcd(N, S - \widehat{S}) = p$.

Since then, other attacks on CRT-RSA have been found, including as recently as last year, when Barthe et al. [3] exposed two new families of fault injections on CRT-RSA: *"almost full" linear combinations of p and q*, and *"almost full" affine transforms of p or q*. Both target intermediate variables of the Montgomery multiplication algorithm (namely Coarsely Integrated Operand Scanning, or CIOS [29,35]) used to implement the exponentiations of the CRT-RSA

Input : Message M, key (p, q, d_p, d_q, i_q)
Output : Signature $M^d \bmod N$

1 $S_p = M^{d_p} \bmod p$ ▷ Intermediate signature in $\mathbb{Z}_p$
2 $S_q = M^{d_q} \bmod q$ ▷ Intermediate signature in $\mathbb{Z}_q$
3 $S = \mathrm{CRT}(S_p, S_q)$ ▷ Recombination in $\mathbb{Z}_N$
4 **return** S

Algorithm 5: Unprotected CRT-RSA.

computation, and both leads to attacks based on the Lenstra-Lenstra-Lovász (LLL) lattice basis reduction algorithm [33].

B Algorithms

B.1 Regular ECSM

The operations on an elliptic curve are point doubling, point addition, and scalar multiplication which can be built on top of the two first operations. A left-to-right ECSM is already sketched in Alg. 1. But for our analysis, we need to detail exactly how it works internally. This is done in Alg. 6, 7, and 8.

Input : $Q = (X_1 : Y_1 : Z_1) \in E(\mathbb{Z}_n)$
Output : $(X : Y : Z) = 2Q \in E(\mathbb{Z}_n)$

1 **if** Q *is* $\mathcal{O}$ **then return** Q
2 $A = 3(X_1^2 + 2aZ_1(X_1 + Z_1))$
3 $X = 2Y_1Z_1(A^2 - 8X_1Z_1Y_1^2)$
4 $Y = A(12X_1Z_1Y_1^2 - A^2) - 8Z_1^2Y_1^4$
5 $Z = 8Z_1^3Y_1^3$
6 **return** $(X : Y : Z)$

Algorithm 6: Elliptic curve doubling ECDBL(Q, n).

B.2 Test-Free ECSM

In this section we present the same algorithms, in their test-free variants. Actually, "test-free" refers to the absence of point comparison (line 1 of Alg. 6, and lines 1, 2, 3, and 4 of Alg. 7). The test depending on the scalar value (i.e., at line 4 of Alg. 11) is still present as it will always be satisfied at the same time in both computations.

Input : $Q = (X_1 : Y_1 : Z_1), P = (X_2 : Y_2 : Z_2) \in E(\mathbb{Z}_n)$
Output : $(X : Y : Z) = Q + P \in E(\mathbb{Z}_n)$

if Q *is* $\mathcal{O}$ **then return** P
if P *is* $\mathcal{O}$ **then return** Q
if $Q = -P$ **then return** $\mathcal{O}$
if $Q = P$ **then return** $\mathrm{ECDBL}(Q, n)$ ▷ See Alg. 6

$A = Y_2Z_1 - Y_1Z_2$
$B = X_2Z_1 - X_1Z_2$
$C = Z_1Z_2A^2 - (X_1Z_2 + X_2Z_1)B^2$

$X = BC$
$Y = A(X_1Z_2B^2 - C) - Y_1Z_2B^3$
$Z = Z_1Z_2B^3$

return $(X : Y : Z)$

Algorithm 7: Elliptic curve addition $\mathrm{ECADD}(Q, P, n)$.

Input : $P \in E(\mathbb{Z}_n)$, $k > 0$
Output : $Q = [k]P \in E(\mathbb{Z}_n)$

$Q = \mathcal{O}$
for $i = \lceil \log_2 k \rceil - 1, \ldots, 0$ **do**
  $Q = \mathrm{ECDBL}(Q, n)$ ▷ See Alg. 6. Precondition: $Q = [\lfloor k/2^{i+1} \rfloor]P$
  if k_i **then** $Q = \mathrm{ECADD}(Q, P, n)$ ▷ See Alg. 7. Precondition: $Q = [2\lfloor k/2^{i+1} \rfloor]P$
return Q

Algorithm 8: Elliptic curve scalar multiplication with left-to-right algorithm $\mathrm{ECSM}_{\mathrm{L2R}}(P, k, n)$.

Input : $Q = (X_1 : Y_1 : Z_1) \in \mathbb{Z}_n^3$
Output : $(X : Y : Z) \in \mathbb{Z}_n^3$

$A = 3(X_1^2 + 2aZ_1(X_1 + Z_1))$

$X = 2Y_1Z_1(A^2 - 8X_1Z_1Y_1^2)$
$Y = A(12X_1Z_1Y_1^2 - A^2) - 8Z_1^2Y_1^4$
$Z = 8Z_1^3Y_1^3$

return $(X : Y : Z)$

Algorithm 9: Test-free elliptic curve doubling $\text{TF-ECDBL}(Q, n)$.

Input : $Q = (X_1 : Y_1 : Z_1), P = (X_2 : Y_2 : Z_2) \in \mathbb{Z}_n^3$
Output : $(X : Y : Z) \in \mathbb{Z}_n^3$

1 $A = Y_2Z_1 - Y_1Z_2$
2 $B = X_2Z_1 - X_1Z_2$
3 $C = Z_1Z_2A^2 - (X_1Z_2 + X_2Z_1)B^2$
4 $X = BC$
5 $Y = A(X_1Z_2B^2 - C) - Y_1Z_2B^3$
6 $Z = Z_1Z_2B^3$
7 **return** $(X : Y : Z)$

Algorithm 10: Test-free elliptic curve addition TF-ECADD(Q, P, n).

Input : $P \in E(\mathbb{Z}_n)$, $k > 0$
Output : $Q = (X : Y : Z) \in \mathbb{Z}_n^3$

1 $Q = \mathcal{O}$
2 **for** $i = \lceil \log_2 k \rceil - 1, \ldots, 0$ **do**
3 $\quad Q =$ TF-ECDBL(Q, n)
4 $\quad$ **if** k_i **then** $Q =$ TF-ECADD(Q, P, n)
5 **return** Q

Algorithm 11: Test-free elliptic curve scalar multiplication with left-to-right algorithm TF-ECSM$_{\text{L2R}}(P, k, n)$.

C TF-Good Scalars for different ECSM Algorithms

We detail in this section the conditions for a scalar to be TF-good for unregular ECSMs, namely L2R & L2R add-always in Sec. C.1, R2L & R2L add-always in Sec. C.2, a sliding window "Non-Adjacent Form" in Sec. C.3, and a regular ECSM, namely the Montgomery ladder in Sec. C.4.

C.1 L2R and L2R add-always [12, §3.1]

The L2R algorithm has already been given in Alg. 8. The conditions for the scalar k to be TF-good are listed in Definition 6.

The L2R add-always is a more costly variant, protected against simple power attacks. It is given in Alg. 12.

It can be seen in Alg. 12 that the point Q_1 is dummy, hence does not impact the computation, even if conditional branches are not taken. Hence a TF-good scalar with respect to left-to-right add-always and left-to-right share the same conditions.

C.2 R2L and R2L add-always

The R2L ECSM algorithm is described in Alg. 13.

We have the following pre-conditions:

- $Q_0 = [k \mod 2^i]P$: at line 4 of Alg. 13.

Input : $P \in E(\mathbb{Z}_n)$, $k > 0$
Output : $Q = [k]P \in E(\mathbb{Z}_n)$

1 $Q_0 = \mathcal{O}$
2 $Q_1 = P$ ▷ Dummy variable, which can be initialized to whatever value
3 **for** $i = \lceil \log_2 k \rceil - 1, \ldots, 0$ **do**
4 $\quad Q_0 = \text{ECDBL}(Q_0, n)$ ▷ See Alg. 6
5 $\quad Q_{1-k_i} = \text{ECADD}(Q_{1-k_i}, P, n)$ ▷ See Alg. 7
6 **return** Q_0

Algorithm 12: Elliptic curve scalar multiplication with left-to-right add-always algorithm $\text{ECSM}_{\text{L2R-AA}}(P, k, n)$.

Input : $P \in E(\mathbb{Z}_n)$, $k > 0$
Output : $Q = [k]P \in E(\mathbb{Z}_n)$

1 $Q_0 = \mathcal{O}$
2 $Q_1 = P$
3 **for** $i = 0, \ldots, \lceil \log_2 k \rceil - 1$ **do**
4 $\quad$ **if** k_i **then** $Q_0 = \text{ECADD}(Q_0, Q_1, n)$ ▷ See Alg. 7
5 $\quad Q_1 = \text{ECDBL}(Q_1, n)$ ▷ See Alg. 6
6 **return** Q_0

Algorithm 13: Elliptic curve scalar multiplication with right-to-left algorithm $\text{ECSM}_{\text{R2L}}(P, k, n)$.

– $Q_1 = [2^i]P$: at line 5 of Alg. 13.

A scalar k is said TF-good using the right-to-left (R2L) ECSM algorithm if tests in ECADD and ECDBL are not taken.

Regarding ECDBL, this means that input Q_1 is not $\mathcal{O}$ at each iteration i. Hence, for all $0 \leq i < \lceil \log_2 k \rceil$, $ord(P) \not| \, 2^i$.

Regarding ECADD, this means that at every iteration i such that $k_i = 1$:

1. Input Q_0 satisfies Q_0 is not $\mathcal{O}$, thus $ord(P) \not| \, k \mod 2^i$.
2. Input Q_1 satisfies Q_0 is not $\mathcal{O}$, which is already covered by the condition: $\forall 0 \leq i < \lceil \log_2 k \rceil, ord(P) \not| \, 2^i$.
3. Inputs Q_0 and Q_1 satisfy $Q_0 + Q_1$ is not $\mathcal{O}$, that is:

$$ord(P) \not| \, (2^i + (k \mod 2^i)$$
$$\iff ord(P) \not| \, (k \mod 2^{i+1}) \quad \text{(recall that by hypothesis, } k_i = 1)$$

4. Inputs Q_0 and Q_1 satisfy $Q_0 - Q_1$ is not $\mathcal{O}$, i.e., $ord(P) \not| \, (2^i - (k \mod 2^i)$.

Thus, a point P is TF-good if all four conditions are satisfied, for all $0 \leq i < \lceil \log_2 k \rceil$:

1. $ord(P) \not| \, 2^i$,
2. $ord(P) \not| \, (k \mod 2^i)$ if $k_i = 1$,
3. $ord(P) \not| \, (k \mod 2^{i+1})$ if $k_i = 1$,
4. $ord(P) \not| \, 2^i - (k \mod 2^i)$ if $k_i = 1$.

The right-to-left add-always ECSM is similar to Alg. 13, except that a dummy point is added to balance the branch depending on k_i bits. Thus, the notion of TF-good point for R2L and R2L-add-always is the same.

C.3 Left-to-Right sliding window NAF scalar multiplication

A *non-adjacent form* (NAF) of a positive integer k is an expression $k = \sum_{i=0}^{l-1} k_i 2^i$ where $k_i \in \{-1, 0, 1\}$, $k_{l-1} \neq 0$, and no two consecutive digits k_i are nonzero. The *length* of the NAF is l.

Theorem 2 ([21, Theorem 3.29]). *Let k be a positive integer.*

- *k has a unique NAF denoted NAF(k) or $(k_{l-1}, \ldots, k_0)_{NAF}$.*
- *NAF(k) has the fewest nonzero digits of any signed digit representation of k.*
- *l is at most one more than the length of the binary representation of k.*
- *The average density of nonzero digits among all NAFs of length n is approximatively $1/3$.*

We recall the L2R sliding window NAF scalar multiplication in Alg. C.4.

It is hard to provide a closed form expression to decide whether a scalar k is good or bad with respect to the multiplication of a point P by k with ECSM algorithm L2R_wNAF. However, such test can be implemented as an algorithm (see Alg. 15).

The idea behind Alg. 15 is to test the conditions for the test-free version of ECADD and ECDBL to work.

In Alg. 14, the code between lines 1 and 4 precomputes $[i]P$ for $i = 3, 5, \ldots, m$. It can easily be checked that the conditions are enumerated as follows:

- At line 2, $ord(P) \neq 1$;
- At line 4, P' is not $\mathcal{O}$, $[i-4]P, [i-2]P, [i]P$ is not $\mathcal{O}$ (for $i = 3, 5, \ldots, m$).

This is clearly equivalent to have the test implemented at line 4 of Alg. 15.

Then, at line 9 of Alg. 14, the ECDBL calls no test provided the point $Q = [(k_{l-1}, \ldots, k_{i+1})_{\mathrm{NAF}}]P$ to be doubled is not $\mathcal{O}$. This is reflected at line 9 of Alg. 15.

The ECDBL at line 17 has a test similar to that at line 9. The presence of test can be tested by the check at line 16 of Alg. 15. In this test, the notation $2^j(k_{l-1}, \ldots, k_{i+1})_{\mathrm{NAF}}$ is short for $(k_{l-1}, \ldots, k_{i+1}, \underbrace{0, \ldots, 0}_{j \text{ zeros}})_{\mathrm{NAF}}$. Actually, the index j goes up to $i - s + 1$, because this last test is required by the next instructions.

Between lines 18 and 21 of Alg. 14, one ECADD is computed. Notice that it is either an addition or a subtraction. But the tests are the same in $Q \pm P_u$, namely $Q, \pm P_u, Q \pm P_u, Q \mp P_u$ is $\mathcal{O}$. One needs to check that:

- P_u is not $\mathcal{O}$, which has already been done at line 4 of Alg. 15.
- Q is not $\mathcal{O}$, which is the test for $j = i - s + 1$ at line 16 of Alg. 15.

Input : $P \in E(\mathbb{Z}_n)$, $w \geq 2$, $k = (k_{l-1}, \ldots, k_0)_{\mathrm{NAF}}$
Output : $Q = [k]P \in E(\mathbb{Z}_n)$

$m \leftarrow 2(2^w - (-1)^w)/3 - 1$
$P' \leftarrow \mathrm{ECDBL}(P, n)$ ▷ See Alg. 6
for $i = 3$ ***to*** m ***by*** 2 **do**
  $P_i \leftarrow \mathrm{ECADD}(P_{i-1}, P')$ ▷ See Alg. 7
$Q \leftarrow P$
$i \leftarrow l - 2$
while $i \geq 0$ **do**
  if $k_i = 0$ **then**
    $Q = \mathrm{ECDBL}(Q, n)$ ▷ See Alg. 6
    $i \leftarrow i - 1$
  else
    $s \leftarrow \max(i - w + 1, 0)$
    while $k_s = 0$ **do**
      $s \leftarrow s + 1$
    $u \leftarrow (k_i, \ldots, k_s)_{\mathrm{NAF}}$
    for $j = 1$ ***to*** $i - s + 1$ **do**
      $Q \leftarrow \mathrm{ECDBL}(Q, n)$ ▷ See Alg. 6
    if $u > 0$ **then**
      $Q \leftarrow \mathrm{ECADD}(Q, P_u, n)$ ▷ See Alg. 7
    if $u < 0$ **then**
      $Q \leftarrow \mathrm{ECADD}(Q, -P_{-u}, n)$ ▷ See Alg. 7
    $i \leftarrow s - 1$
return Q

Algorithm 14: Elliptic curve scalar multiplication with left-to-right sliding window NAF algorithm $\mathrm{ECSM}_{\mathrm{L2R\_wNAF}}(P, k, n)$.

Input : $P \in E(\mathbb{Z}_n)$, $w \geq 2$, $k = (k_{l-1}, \ldots, k_0)_{\text{NAF}}$
Output : True if k is TF-good, or False otherwise

$m \leftarrow 2(2^w - (-1)^w)/3 - 1$
$P' \leftarrow \text{ECDBL}(P, n)$ ▷ See Alg. 6
for $i = 3$ ***to*** m ***by*** 2 **do**
  if $ord(P) \mid i$ **then return** False
$Q \leftarrow P$
$i \leftarrow l - 2$
while $i \geq 0$ **do**
  if $k_i = 0$ **then**
    if $ord(P) \mid (k_{l-1}, \ldots, k_{i+1})_{NAF}$ **then return** False
    $i \leftarrow i - 1$
  else
    $s \leftarrow \max(i - w + 1, 0)$
    while $k_s = 0$ **do**
      $s \leftarrow s + 1$
    for $j = 0$ ***to*** $i - s + 1$ **do**
      if $ord(P) \mid 2^j(k_{l-1}, \ldots, k_{i+1})_{NAF}$ **then return** False
    if $ord(P) \mid (k_{l-1}, \ldots, k_{i+1}, \pm k_i, \ldots, \pm k_s)_{NAF}$ **then return** False
    $i \leftarrow s - 1$
return True

Algorithm 15: Test whether a scalar k is TF-good or TF-bad with respect to the ECSM of a point P using the left-to-right sliding window NAF algorithm $\text{ECSM}_{\text{L2R\_wNAF}}$.

– As already mentioned, checking an addition or a subtraction imply the same tests, which can be summarized as: $ord(P) \mid 2^{i-s+1}(k_{l-1}, \ldots, k_{i+1})_{\text{NAF}} + \pm(k_i, \ldots, k_s)_{\text{NAF}}$, as is tested at line 17 of Alg. 15.

C.4 Montgomery Powering Ladder

The Montgomery powering ladder is a regular ECSM, described in Alg. 16.

Input : $P \in E(\mathbb{Z}_n)$, $k > 0$
Output : $Q = [k]P \in E(\mathbb{Z}_n)$

1 $Q_0 = \mathcal{O}$
2 $Q_1 = P$
3 **for** $i = \lceil \log_2 k \rceil - 1, \ldots, 0$ **do**
4 $\quad Q_{1-k_i} = \text{ECADD}(Q_0, Q_1, n)$ ▷ See Alg. 7
5 $\quad Q_{k_i} = \text{ECDBL}(Q_{k_i}, n)$ ▷ See Alg. 6
6 **return** Q_0

Algorithm 16: Elliptic curve scalar multiplication with Montgomery powering ladder algorithm $\text{ECSM}_{\text{Mont}}(P, k, n)$.

We have the following pre-conditions:

– $Q_0 = [\lfloor k/2^{i+1} \rfloor]P$ and $Q_1 = [\lfloor k/2^{i+1} \rfloor + 1]P$: at line 4 of Alg. 16.
– $Q_{k_i} = [\lfloor k/2^{i+1} \rfloor + k_i]P$: at line 5 of Alg. 16.

A scalar k is said TF-good using the Montgomery powering ladder algorithm if tests in ECADD and ECDBL are not taken.

Regarding ECADD, this means that at every iteration i such that $\lceil \log_2 k \rceil > i \geq 0$:

1. Input Q_0 satisfies Q_0 is not $\mathcal{O}$, thus $ord(P) \nmid \lfloor k/2^{i+1} \rfloor$.
2. Input Q_1 satisfies Q_1 is not $\mathcal{O}$, thus $ord(P) \nmid \lfloor k/2^{i+1} \rfloor + 1$.
3. Inputs Q_0 and Q_1 satisfy $Q_0 \neq -Q_1$, i.e., $Q_0 + Q_1 = [2\lfloor k/2^{i+1} \rfloor + 1]P$ is not $\mathcal{O}$, thus $ord(P) \nmid 2\lfloor k/2^{i+1} \rfloor + 1$. But,
$$2\lfloor k/2^{i+1} \rfloor + 1 = \begin{cases} \lfloor k/2^i \rfloor + 1 & \text{if } k_i = 0, \\ \lfloor k/2^i \rfloor & \text{if } k_i = 1, \end{cases}$$
hence this condition is already covered by the first two ones.
4. Inputs Q_0 and Q_1 satisfy $Q_0 \neq Q_1$. In Montgomery powering ladder, $Q_1 - Q_0 = P$ at every iteration i, thus we must have $P \neq \mathcal{O}$, i.e., $ord(P) \neq 1$, which is also already covered by the two first conditions.

Regarding ECDBL, this means that input Q_{k_i} is not $\mathcal{O}$ at each iteration i. This is equivalent to $ord(P) \nmid \lfloor k/2^{i+1} \rfloor + k_i$, which is also already covered by the two first conditions of ECADD.

So, to summarize, a scalar is TF-good when used in conjunction with a Montgomery powering ladder if, for all i such that $\lceil \log_2 k \rceil > i \geq 0$:

1. $ord(P) \nmid \lfloor k/2^{i+1} \rfloor$, and
2. $ord(P) \nmid \lfloor k/2^{i+1} \rfloor + 1$.

D Details on the Existing Countermeasures

D.1 BOS Combined Curve and Point

Here we detail the computation behind line 2 of Alg. 2. For the combined curve $E(\mathbb{Z}_{pr})$ (we denote A_n and B_n the equation parameters of the curve on $\mathbb{Z}_n$):
- $A_{pr} = \mathrm{CRT}(A_p, A_r)$,
- $B_{pr} = \mathrm{CRT}(B_p, B_r)$.

For the point P_{pr}:
- $X_{pr} = \mathrm{CRT}(X_p, X_r)$,
- $Y_{pr} = \mathrm{CRT}(Y_p, Y_r)$,
- $Z_{pr} = \mathrm{CRT}(Z_p, Z_r)$.

Here, $\mathrm{CRT}(U_a, U_b)$ denotes the CRT recombination in $\mathbb{Z}_{ab}$ of $U_a \in \mathbb{F}_a$ and $U_b \in \mathbb{F}_b$.

It is possible to avoid these computations by choosing $U_r = U_p \bmod r$ in order to have $U_{pr} = U_p$ for all the variables listed above. However, it will be less secure than choosing security-optimized U_r values. Indeed, recall comments made about Tab. 1.

D.2 BOS Numerical Example of Incorrect Result

Here we give a toy example for which BOS returns an incorrect result. We chose an unrealistic, very small r as it allows to verify the computations quickly.

The nominal elliptic curve $E(\mathbb{F}_p)$ (P-192) has the following parameters:

$$\begin{aligned} p &= \texttt{0xfffffffffffffffffffffffffffffffeffffffffffffff}, \\ A &= p - 3, \\ B &= \texttt{0x64210519e59c80e70fa7e9ab72243049feb8deecc146b9b1}, \\ x_P &= \texttt{0x188da80eb03090f67cbf20eb43a18800f4ff0afd82ff1012}, \\ y_P &= \texttt{0x07192b95ffc8da78631011ed6b24cdd573f977a11e794811}. \end{aligned}$$

The small curve $E(\mathbb{F}_r)$:

$$\begin{aligned} r &= \texttt{0x7}, \\ A_r &= \texttt{0x4}, \\ B_r &= \texttt{0x1}, \\ x_{P_r} &= \texttt{0x6}, \\ y_{P_r} &= \texttt{0x4}. \end{aligned}$$

The combined curve $E(\mathbb{Z}_{pr})$ obtained as explained in Sec. D.1:

$$\begin{aligned} pr &= \texttt{0x6fffffffffffffffffffffffffffffffffffffffffffffff9}, \\ A_{pr} &= \texttt{0x6fffffffffffffffffffffffffffffffffffffffffffffff6}, \\ B_{pr} &= \texttt{0x264210519e59c80e70fa7e9ab72243047feb8deecc146b9af}, \\ x_{P_{pr}} &= \texttt{0x3188da80eb03090f67cbf20eb43a187fdf4ff0afd82ff100f}, \\ y_{P_{pr}} &= \texttt{0x307192b95ffc8da78631011ed6b24cdd273f977a11e79480e}. \end{aligned}$$

Given these parameters, Alg. 2 is correct for all k such that $0 < k \leq 8$. Actually, $P_r = (x_{P_r} : y_{P_r} : 1)$ is a point of order 8 on $E(\mathbb{F}_r)$, hence $[8]P_r = \mathcal{O}$.

However, the results start to be incorrect when $k > 8$. For example:

- $[9]P$ on $E(\mathbb{F}_p)$ is equal to

$$(\texttt{0x818a4d308b1cabb74e9e8f2ba8d27c9e1d9d375ab980388f},$$
$$\texttt{0x1d1aa5e208d87cd7c292f7cbb457cdf30ea542176c8e739}),$$

  and no conditional tests are satisfied during the computation.
- $Q = [9]P_r$ on $E(\mathbb{F}_r)$ is equal to $(\texttt{0x6}, \texttt{0x4})$, and one conditional test is satisfied, namely Q is $\mathcal{O}$ (line 1 of Alg. 6).
- $[9]P_{pr}$ on $E(\mathbb{Z}_{pr})$ is equal to

$$(\texttt{0x3818a4d308b1cabb74e9e8f2ba8d27c9b1d9d375ab980388c},$$
$$\texttt{0x401d1aa5e208d87cd7c292f7cbb457cdb30ea542176c8e735}),$$

  which matches $[9]P$ on $E(\mathbb{F}_p)$ modulo p, but is equal to $(\texttt{0x0}, \texttt{0x0}) \neq (\texttt{0x6}, \texttt{0x4})$ modulo r.

As a result, error will be returned while no error nor fault attacks actually occurred.

D.3 BV Combined Curve and Point

Here we detail the computation behind line 2 of Alg. 3.

In their paper [2], Baek and Vasyltsov use the following curve equation (in Jacobian projective coordinate) for the curve $E(\mathbb{F}_p)$: $Y^2Z = X^3 + AXZ^4 + BZ^6$. To obtain the combined curve $E'(\mathbb{Z}_{pr})$, a value B' is computed first: $B' = y^2 + py - x^3 - ax \bmod pr$, where $(x : y : 1)$ are the Jacobian projective coordinates of P. Then, the equation of the curve $E'(\mathbb{Z}_{pr})$ is: $Y^2Z + pYZ^3 = X^3 + AXZ^4 + B'Z^6$.

Notice that the check at line 4 of Alg. 3 is based on this Jacobian projective coordinate equation[11], only taken modulo r.

The same point P is used.

E Complexity of Modular Inverse in Direct Product

The modular inverse of a number can be efficiently obtained thanks to an extended Euclid algorithm [36, Algorithm 2.107 at §2.4, p. 67]. The complexity of this algorithm is quadratic in the size in bits of the modulo (i.e., it is $O(\log_2(p))$ when the ring is $\mathbb{Z}_p$), like the modular multiplication (see [36, Table 2.8 in Chap. 2, page 84]). However, in practice, for moduli of cryptographic size (i.e., $192 \leq \log_2(p) \leq 521$ for ECC, see [40]) the duration of a division lasts from 4 to 10 times the duration of a multiplication[12].

[11] In the original paper [2], there is a typo in Alg. 2 and the equation reads $Y^2 + pYZ^3 = X^3 + AXZ^4 + B'Z^6$, missing a Z.

[12] As benchmarked by OpenSSL version 1.0.1f `BN_mod_mul` and `BN_mod_inverse` functions.

Prop. 6 shows that divisions can also be implemented efficiently in $\mathbb{Z}_{pr}$, provided the division exists. If not, an exponentiation $z \mapsto z^{p-2}$ is necessary. Assuming that the binary representation of p consists of as many ones as zeros and that the exponentiation is done with a double-and-add algorithm, the cost of $z \mapsto z^{p-2}$ is about $\frac{3}{2}\lfloor \log_2 p \rfloor$, that is 288 multiplications when p is a 192-bit prime number.

However, multiples of r occur only with probability $\frac{1}{r}$ in computations. Thus, an upper-bound for the expected overhead is $(10\times(1-\frac{1}{r})+384\times\frac{1}{r})/10 \approx 1+10^{-8}$ when r is a 32 bit number, which is negligible in practice.

F Theoretical Upper-Bound for #roots

It is interesting to study the theoretical upper bound on the number of roots in practical cases. Leont'ev proved in [34] that if P is a random polynomial in $\mathbb{F}_p$ then $\#\text{roots}(P) \sim \text{Poisson}(\lambda = 1)$, i.e., $\mathbb{P}(\#\text{roots}(P) = k) = \frac{1}{ek!}$. In the case of $\Delta P \bmod r$, we know that there is always at least one root, when $\widehat{x_1} = x_1$, so we can rewrite $\Delta P(\widehat{x_1}) = P(\widehat{x_1}) - P(x_1) = R(\widehat{x_1}) \cdot a(\widehat{x_1} - x_1)$, where a is some constant, and R is indeed a random polynomial of degree $r-2$, owing to the modular reduction of ΔP by r. So we know that $\#\text{roots}(\Delta P) = 1 + \#\text{roots}(R)$, hence $\mathbb{P}(\#\text{roots}(\Delta P) = k) = \mathbb{P}(\#\text{roots}(R) = k-1)$, which is 0 if $k = 0$ and $\frac{1}{e(k-1)!}$ otherwise. We want the maximum value of k which has a "plausible" probability, let us say that is 2^{-p}, e.g., 2^{-256}. Since the values of a Poisson distribution of parameter $\lambda = 1$ are decreasing, we are looking for k such that: $\mathbb{P}(\#\text{roots}(R) = k-1) = \frac{1}{e(k-1)!} \leq 2^{-256}$. This would suggest that $k \gtrsim 58$.

This result means that $\mathbb{P}_{\text{n.d.}}$ is predicted to be at most $\frac{57}{r}$, with r being at least a 32-bit number, i.e., that $\mathbb{P}_{\text{n.d.}}$ is at maximum $\approx 2^{-26}$, and that this worst-case scenario has a probability of $\approx 2^{-256}$ of happening, in theory.

Remark 3. Note that we do not take into account the probability of TF-bad k. However, the probability of k being TF-bad can be bounded with respect to a point $P \in E(\mathbb{F}_r)$, as explained in Prop. 5 (Sec. 5.2).

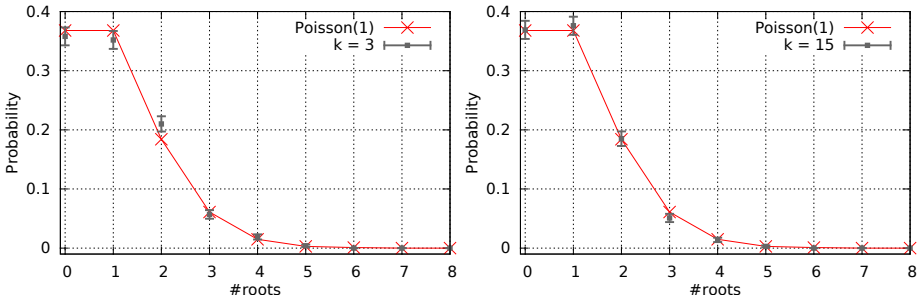


Fig. 4: #roots probability for ECSM $[k]G$.

Fig. 4 shows typical number of roots (obtained with SAGE) for practical cases in ECC, and compare them to the theoretical predictions. In this figure,

we chose values of k of the form 2^j-1, which maximize the number of operations, and thus the size and degree of the resulting ΔP polynomials. For each value of k, we expressed the polynomial ΔP corresponding to the ECSM $[k]G$, and did so for a thousand random G. We then plotted for $i=0$ to 8 the number of $[k]G$ for which $\#\text{roots}(\Delta P)=i+1$ divided by 1000, that is the estimated probability $\hat{\mathbb{P}}(\#\text{roots}(\Delta P)=i+1)$. Let us denote by Z the Boolean random variable which is equal to one if ΔP has a $(i+1)$ roots, and zero otherwise. Our estimation of $\hat{\mathbb{P}}(\#\text{roots}(\Delta P)=i+1)$ is thus the expectation of $\frac{1}{1000}\sum_{j=1}^{1000} Z_j$. This random variable follows a binomial distribution, of mean $p=\mathbb{P}(\#\text{roots}(\Delta P)=i+1)$ and variance $p(1-p)/1000$. The later values are used for the error bars ($[p-\sqrt{p(1-p)/1000}, p+\sqrt{p(1-p)/1000}]$).

The two graphs in Fig. 4 correspond to two corner-cases:

1. $k=3=(11)_2$: the number of roots is small because the polynomial degree is small (it is 13). (recall that $\#\text{roots}(P)$ cannot exceed the degree of P).
2. $k=15=(1111)_2$: the number of roots is also small, but this times because the result of Leont'ev applies. Indeed, the degree is 7213, thus the polynomial is much more random-looking.

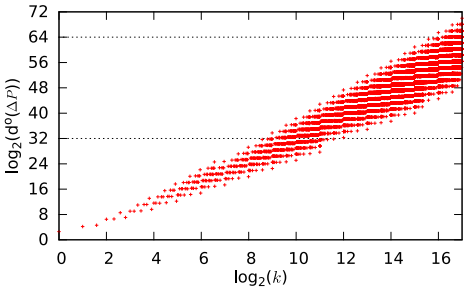


Fig. 5: Degree of the polynomial ΔP against the value of k (in log-log scale).

Actually, it is computationally hard to count the roots of polynomials of degree greater than 7213. But it can be checked that the degree of the polynomials is growing exponentially with k. This is represented in Fig. 5, where we see that the degree is about equal to $k^{3.25}$ (of course, when k has a large Hamming weight, as in $(11\ldots1)_2$, the degree is larger than when k is hollow, as in $(10\ldots0)_2$). In particular, the polynomial ΔP reaches degree 2^{32} (resp. 2^{64}) when k has about 10 (resp. 18) bits. Thus, modulo r (recall Eqn. (1)), the polynomial ΔP has maximal degree as long as the fault is injected before the last 10 (resp. 18) elliptic curve operations when r fits on 32 bits (resp. 64 bits).

G Examples of $\mathbb{P}_{\text{n.d.}}$

Example 1 ($\mathbb{P}_{n.d.}$ for CRT-RSA). From Thm. 1's proof, we can derive that for proven CRT-RSA countermeasures such as [1,42,38], we have $\mathbb{P}_{\text{n.d.}}=\frac{1}{r}$.
Indeed, in CRT-RSA, the computation mainly consists of two exponentiations. In an exponentiation, ΔP takes on the form $\widehat{m}^k\cdot m^{d-k}-m^d=(\widehat{m}^k-m^k)\cdot m^{d-k}$. Assuming the message $m\neq 0$, we have $\#\text{roots}(\Delta P)=1$ (that is $\widehat{m}=m \mod r$),

hence a non-detection probability of $\frac{1}{r}$ (in the case RSA is computed with CRT). Otherwise, after the Garner recombination [17] ΔP is of the form $m^{d_q} + q \cdot (i_q \cdot (m^{d_p - k}\widehat{m}^k - m^{d_q}) \bmod p) - m^{d_q} + q \cdot (i_q \cdot (m^{d_p} - m^{d_q}) \bmod p) = q \cdot (i_q \cdot (m^{d_p - k}\widehat{m}^k - m^{d_p}))$, if the fault is on the p part; or $m^{d_q - k}\widehat{m}^k + q \cdot (i_q \cdot (m^{d_p} - m^{d_q - k}\widehat{m}^k) \bmod p) - m^{d_q} + q \cdot (i_q \cdot (m^{d_p} - m^{d_q}) \bmod p) = (m^{d_q - k}\widehat{m}^k - m^{d_q}) + q \cdot (i_q \cdot (m^{d_q - k}\widehat{m}^k - m^{d_q}))$, if it is on the q part.
We conclude that #roots(ΔP) is still 1 in both cases and thus that $\mathbb{P}_{\text{n.d.}} = \frac{1}{r}$.

It can be noticed that this result had been used in most previous articles dealing with fault protection on CRT-RSA without being formally proved. So, we now formally confirm those results were indeed correct.

Example 2 (Fault non-detection probability greater than $\frac{1}{r}$). Let us assume the computation $P(a, b, c) = (a + b) \cdot (b + c)$. If a single fault strikes b, then the polynomial ΔP is equal to $P(a, \hat{b}, c) - P(a, b, c) \mod r$. Its degree is equal to 2, and has 2 distinct roots provided $b \neq -(a + c)/2 \mod r$, or 1 double root otherwise. Thus, in the general case where the nominal inputs satisfy $b \neq -(a + c)/2 \mod r$ (which occurs also with probability $\frac{1}{r}$), the non-detection probability is $\frac{2}{r}$. Namely, the $2p - 1$ values of $\hat{b} \in \mathbb{Z}_{pr}$ causing an undetected fault are $b + kr$, with $k \in \{1, \ldots, p - 1\}$, and $-(a + c)/2 + lr$, with $l \in \{0, \ldots, p - 1\}$.