KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT INGENIEURSWETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK–ESAT
Kasteelpark Arenberg 10, 3001 Leuven-Heverlee

White-Box Cryptography

Promotor:
Prof. Dr. ir. Bart Preneel

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de ingenieurswetenschappen

door

Brecht WYSEUR

Maart 2009

KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT INGENIEURSWETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK–ESAT
Kasteelpark Arenberg 10, 3001 Leuven-Heverlee

White-Box Cryptography

Jury:
Prof. dr. ir.-arch. Herman Neuckermans, voorzitter
Prof. dr. ir. Bart Preneel, promotor
Dr. Henri Gilbert (Orange Labs)
Prof. dr. ir. Jean-Jacques Quisquater (UCL)
Prof. dr. ir. Vincent Rijmen
Prof. dr. ir. Marc Van Barel
Prof. dr. ir. Joos Vandewalle

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de ingenieurswetenschappen

door

Brecht WYSEUR

U.D.C. 681.3*E4 Maart 2009

D/2009/7515/23

ISBN 978-94-6018-040-8

For those who keep trying,
failure is temporary
– Frank Tyger

Acknowledgments

This dissertation is the result of a long process to which many people were directly or indirectly involved. Therefore, I would like to take this opportunity to thank everyone who has supported me during the past years.

First, I would like to express my gratitute to Prof. Bart Preneel for giving me the opportunity to pursue a Ph.D. at COSIC. For this guidance and advice during the past years, and carefully reading and correcting this dissertation. And for the opportunity to explore my personal interests and set my own objectives. I want to thank Dr. Henri Gilbert, Prof. Jean-Jacques Quisquater, Prof. Vincent Rijmen, Prof. Marc Van Barel, and Prof. Joos Vandewalle for kindly accepting to be members of the jury, and Prof. Herman Neuckermans for chairing it.

When I started working at COSIC, colleagues had to perform some kind of speech-cryptanalysis in order to understand me. Fortunately, in time the communication improved and while cosic was growing fast, many colleagues became friends. The social events that we organized were a nice addition to support friendship amongst colleagues. Hence, I would like to thank the past and current COSIC members for the special atmosphere, including An, Antoon, Bartek, Benedikt, Christophe, Christopher, Danny, Dave, Elmar, Elke, Frederik, George, Gregory, Jens, Joe, Klaus, Koen, Kåre, Markulf, Miroslav, Nele, Nessim, Nicky, Norbert, Robert, Pim, Roel, Saartje, Sebastian, Sebastiaan, Souradyuti, Stefaan, Stefan, Svetla, Thomas, and Wouter. In particular, I would like to thank Jan Cappaert, Dries Schellekens, and Karel Wouters for all those years of sharing and moving offices together throughout the entire ESAT building, and with Elena Andreeva and Yoni De Mulder, for the good office atmosphere.

The fact that I can present this work to you, has only been possibly by virtue of people that are (mostly in a healthy way) obsessed by Mathematics, and have divulged this enthousiasm to many students. In particular, I want to thank Marc Platteeuw, Prof. Paul Igodt, Prof. Jan Denef, and Prof. Igor Semaev, for their joy of Mathematics they have shared with me.

Péla Noë deserves a big thank you for many reasons. Not only for practical matters such as paperwork, but also because she is someone whom you can always talk to. And Elvira Wouters for the immens administrative work she is able to cope with. Dank!

Graag dank ik mijn ma en pa voor de kansen die ze me geboden hebben. Mijn zussen Liselot en Julie om het met me uit te houden. Ik geef toe dat ik misschien niet altijd de gemakkelijkste broer of zoon ben geweest, maar toch zijn ze me blijven aanmoedigen. Alsook aan m'n ganse familie. Bedankt voor de blijvende steun! Mijn vriendin Lut verdient ook een woordje dank voor haar liefde, steun en de tijd die ze me gaf om dit werk te voltooien.

Vriendschap geeft telkens een impuls om terug verder te gaan. Ik wens dan ook tal van vrienden te bedanken voor de vele tijd die we samen hebben beleefd. Een extra dank u aan Lies Verdru en Thomas rotté, voor hun gastvrije ontvangst elke zondagavond en onze uitstapjes samen. Alle vrienden hier bedanken wordt moeilijk, maar toch een woordje aan Dries en Inge, Luc en Ilse, Stefaan, Joris (aka Sjors), Koen en Melanie, Lieven en Delphine, Anneleen, Bruno, Elke, Koen, Leander, Wim. Vrienden van vroeger en tijdens het studentenleven. Dank aan Lies, Inge, Yoni, en Karel om het aantal (dt-) fouten in deze tekst te reduceren.

Ut Vivat, crescat et floreat studentenclub Moeder Baekelandt! Dank aan haar (oud-)leden, collega pro-senioren en praesidiumleden voor de mogelijkheden die aangeboden werden om even aan het Ph.D leven te ontsnappen met een pint aan de toog.

Last but not least, I would also like to acknowledge the K.U.Leuven and the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT), for funding my research work.

Brecht Wyseur
March 2009

Abstract

This thesis studies the topic of *white-box cryptography* (WBC), which focusses on software implementations of cryptographic primitives (such as encryption schemes). Traditionally, cryptographic primitives are designed to protect data and keys against *black-box* attacks. In such a context, an adversary has knowledge of the algorithm and may examine various inputs to and outputs from the system, but has no access to the internal details of the execution of a key instantiated primitive. In contrast, the goal of white-box implementations is to provide a degree of robustness against attacks from the execution environment. In such an environment, an adversary has unrestricted access to the software implementation.

The main part of this dissertation covers the security assessment of white-box implementations. This contribution is two-fold: we study practical white-box techniques and perform a theoretical study. First, a study is conducted on the practical white-box implementations of DES and AES encryption algorithms, which includes their cryptanalysis. Subsequently, generic cryptanalysis results are described, which opens a discussion towards white-box design strategies.

Since no formal definitions of white-box cryptography were presented before and the proposed white-box implementations did not come with any proof of security, we initiate a study towards a theoretical model for white-box cryptography. The study on formal models of obfuscation and provable security leads to a definition where we capture the security requirements of WBC defined over some cryptographic scheme and a security notion. This new theoretical model provides a context to investigate the security of white-box implementations, which leads to a number of positive and negative results.

Considering the practical interest of research in WBC, we conclude with an overview of a selection of applications and related research fields that might benefit from and contribute to this research topic.

Samenvatting

Deze thesis bestudeert het onderwerp *white-box cryptografie* (WBC), dat zich op software-implementaties van cryptografische primitieven (zoals vercijferingsschema's) concentreert. Traditioneel worden cryptografische primitieven ontworpen om gegevens en sleutels te beschermen tegen *black-box* aanvallen. Hierbij heeft de aanvaller kennis van het algoritme en kan hij de invoer naar en de uitvoer van het primitief bestuderen. Maar hij heeft geen zicht op de interne werking van een sleutel-geïnstantiëerd primitief tijdens de uitvoering (zwarte doos). In deze thesis beschouwen we een aanvalsmodel waarbij de aanvaller onbeperkte toegang heeft tot de software-implementatie: het white-box model. Het doel van white-box implementaties is om in een dergelijke context een bepaald niveau van bescherming te bekomen.

Het belangrijkste deel van deze thesis behandelt het nagaan van de veiligheid van white-box implementaties. Deze bijdrage is tweezijdig. Allereerst worden de praktische white-box implementaties van DES en AES vercijferingsalgoritmen beschreven, en hun cryptanalyse voorgesteld. Deze resultaten worden verder uitgebreid naar generische aanvallen, wat het pad opent naar nieuwe technieken.

Aangezien nog geen formele definities van white-box cryptografie voorgesteld werden en de praktische white-box implementaties zonder bewijs van veiligheid voorzien zijn, dringt zich een studie op naar het formeel definiëren van white-box cryptografie. Dit is de tweede grote bijdrage. De studie van theoretische obfuscatie en bewijsbare veiligheid resulteert in een definitie van WBC, waarbij we de veiligheidsvereisten omvatten voor een bepaald cryptografisch primitief met geassociëerde veiligheidsnotie. Dit nieuw theoretisch model levert een context op, waarin de veiligheid van white-box implementaties bestudeerd kan worden, en leidt tot een aantal positieve en negatieve resultaten.

Gezien de praktische mogelijkheden die kunnen voortvloeien uit WBC, beëindigen we deze thesis met een overzicht van een selectie van toepassingen en verwante onderzoeksdomeinen, aan welke dit onderzoek kan toe bijdragen.

Contents

List of Figures

List of Tables

List of Abbreviations

AES	Advanced Encryption Standard
AT	Affine Transformation
BC	Block Cipher
CBC	Cipher-block chaining
CCA	Chosen Ciphertext Attack
CED	Computing on Encrypted Data
CPA	Chosen Plaintext Attack
CPU	Central Processing Unit
DES	Data Encryption Standard
DRM	Digital Rights Management
FTG	Find Then Guess
GPS	Global Positioning System
IND	Indistinguishability
LUT	Lookup Table
OT	Oblivious Transfer
SFE	Secure Function Evaluation
SLT	Subsitution-Linear Transformation (cipher)
SN	Security Notion
SPN	Substitution-Permutation Network (cipher)
TM	Turing Machine
TRS	Tamper Resistant Software
VBBP	Virtual Black Box Property
WBC	White-Box Cryptography
WBP	White-Box Property

List of Notations

Functions, Sets, Elements

E_k	Encryption with key $k \in \mathcal{K}$
D_k	Decryption with key $k \in \mathcal{K}$
$\mathcal{K}$	Key space
$k \in \mathcal{K}$	Key
$\mathcal{P}, \mathcal{C}$	Plaintext space, ciphertext space
$c \in \mathcal{C}; m, p \in \mathcal{P}$	A ciphertext; message or plaintext
$m, p \in \mathcal{P}$	Message or plaintext
$\mathrm{GF}(q)$	Finite field with q elements
$\mathrm{GF}(q)^n$	n-dimensional vector space of $\mathrm{GF}(q)$
$\{0,1\}^n$	Bit-string of length n, i.e., any piece of digital data ($\{0,1\}^*$ indicates an arbitrary length).
S	A substitution operation or S-box
L	A lookup table
$\mathcal{R}$	Random oracle
$a \leftarrow \mathcal{A}$	Selection of an element a from a set $\mathcal{A}$ (under some distribution)
$Q[q]$	Instantiation of algorithm Q by $q \in \mathcal{K}$
$A^P(x)$	Process A that computes on input x, with oracle access to P
$\mathcal{O}$ – $\mathcal{O}(P)$	Obfuscator – obfusation of P
$\times_{i=1}^{n} N_i$	Product space $N_1 \times N_2 \times \cdots \times N_n$

Basic Operators

$\oplus$	the exclusive-or operation or addition modulo 2
$\oplus_c$	a function that adds c to the input (modulo 2) $\oplus_c(x) = x \oplus c$
$a\|\|b$	the concatenation of the strings/values a and b
$\|a\|$	absolute value of a, or bit-length of a (depending on the context)
$\|A\|$	Size of the set A
$\lceil a \rceil$	The smallest integer larger than or equal to a
$g \circ f$	The composition of the functions f and g $(g \circ f)(x) = g(f(x))$

White-Box Cryptografie

Nederlandse samenvatting

Hoofdstuk 1: Inleiding

Het gebruik van computers in ons dagelijkse leven neemt toe. Daarbij denken we niet enkel aan onze huishoudelijke elektronica (computer, gsm, televisie), maar onder meer ook aan de computergestuurde besturing van onze wagens, regeling van het elektriciteitsnetwerk, online bankrekening, en internet. Steeds meer steunt onze maatschappij op computerinfrastructuur en het uitwisselen en verwerken van gegevens. In geïndustrialiseerde landen steunen quasi alle bedrijven voor hun dagelijkse werking direct of indirecte op computers en software. Het industriële en huishoudelijke gebruik wordt verder gestimuleerd door de indrukwekkende expansie van internet en (draadloze) communicatienetwerken. Door het open karakter van tal van netwerken kan iedereen met verschillende toestellen hieraan connecteren.

Om gebruik te kunnen maken van de nieuwe opportuniteiten die computersystemen bieden, is er nood aan methodes die de communicatie en het verwerken van gegevens veilig stellen. Cryptologie vormt hierbij een hoeksteen en omvat het ontwikkelen van algoritmes (cryptografie) en het breken van deze algoritmes (cryptanalyse), met als centraal doel het beveiligen van gegevens. Voorbeelden van dergelijke algoritmes zijn onder andere gekend als vercijferingsalgoritmes en digitale handtekeningen.

Aanvankelijk werd cryptologie toegepast voor het beschermen van militaire en diplomatieke gegevens. Tegenwoordig wordt ze gebruikt voor het beveiligen van netwerken, computersystemen en allerhande applicaties, zowel voor bedrijven als voor particulieren. Vooral het beveiligen van softwareapplicaties is een recent fenomeen. De nood naar beveiliging hiervan wordt versterkt door een toenemende trend van het gebruik van complexe softwareapplicaties met strenge beveiligingsvereisten. Denk daarbij onder andere aan online bankprogramma's,

online spelletjes waar digitale valuta geassocieerd worden met reële valuta, en digitale mediasystemen. Bovendien wordt dergelijke software vaak uitgevoerd op onbetrouwbare systemen. Onbetrouwbaar vanuit het standpunt van de gebruiker (bv. door de aanwezigheid van malware), of vanuit het standpunt van een leverancier die niet wenst dat een gebruiker van de software bijvoorbeeld ongeoorloofd digitale valuta kan aanmaken of digitale media kan kopiëren en verspreiden.

Dit brengt ons bij de doelstelling van deze thesis. Cryptografische primitieven werden ontwikkeld om uitgevoerd te worden op betrouwbare computerplatformen. Het principe van Kerckhoffs staat hierin centraal, en stelt dat de veiligheid van cryptografische algoritmes enkel berust op de geheimhouding van sleutelinformatie, terwijl het algoritme zelf publieke kennis is. Ze worden echter steeds meer gebruikt in softwareapplicaties die uitgevoerd op onbetrouwbare systemen, wat problemen stelt bij de geheimhouding van sleutelinformatie. De doelstelling is na te gaan hoe cryptografische primitieven zoals vercijferingsalgoritmes veilig kunnen worden geïmplementeerd, zodat de aanvaller geen sleutelinformatie kan bekomen ondanks het feit dat die aanvaller volledige toegang heeft tot het computerplatform of zelfs de software-implementatie van het vercijferingsalgoritme volledig ter beschikking heeft. Dit onderzoek wordt *white-box* (witte doos) cryptografie genoemd, wat de context van volledige open toegang schetst in tegenstelling tot een zwarte doos.

In het eerste hoofdstuk van deze thesis leiden we het onderzoeksdomein van de cryptografie in en schetsen we de probleemstelling. We illustreren dit aan de hand van 2 voorbeelden van mogelijke aanvallen op software implementaties. Een eerste voorbeeld omschrijft een aanval die het geheugen uitleest tijdens de uitvoering van een cryptografisch algoritme. Vermits er sleutelinformatie gebruikt wordt tijdens een vercijferingsproces, zal sleutelinformatie in het geheugen opgeslagen worden. Dergelijke informatie ziet er typisch meer willekeurig (hoge entropie) uit dan uitvoerbare code (lage entropie), en kan door middel van een entropieonderzoek van het geheugen gelokaliseerd worden. Het blijkt moeilijk zich hiertegen te beschermen. Een typische tegenmaatregel is het opbreken van sleutelinformatie in kleinere stukken en op uiteenliggende locaties in het geheugen opslaan, of ervoor te zorgen dat telkens maar een klein deel sleutelinformatie in het geheugen staat. Tegen een geschoolde, gemotiveerde aanvaller halen deze beschermingsmaatregelen echter niet veel uit.

Een tweede aanval beschrijft een aanval op de software implementatie zelf, waarbij een herkenbaar deel bewust overschreven wordt. Bij vele populaire cryptografische algoritmes kan het overschrijven van een stuk (gecompileerde) code leiden tot het weergeven van stukken sleutelinformatie bij uitvoering van de implementatie.

Deze thesis levert een bijdrage tot mogelijke oplossingen voor deze uitda-

ging. Nadat in hoofdstuk 1 een context is omschreven, worden in hoofdstuk 2 de cryptografische primitieven omschreven die we veilig wensen te implementeren. We schetsen er ook een overzicht van bestaande aanvallen op cryptografische primitieven in traditionele aanvalsmodellen. In het onderzoek naar white-box cryptografie onderscheiden we drie belangrijke aspecten. We behandelen deze elk in een hoofdstuk van deze thesis.

In hoofdstuk 3 presenteren we technieken die recent werden voorgesteld voor het veilig implementeren van cryptografische primitieven in software. Ons onderzoek omvat een grondige analyse van deze technieken, en leidt tot het opstellen van vereisten in het ontwerp van nieuwe vercijferingsalgoritmes. In hoofdstuk 4 schetsen we een theoretisch raamwerk voor white-box cryptografie, waarbinnen we de theoretische grenzen van wat wel en niet mogelijk is afzoeken. Hoofdstuk 5 verdiept zich op de toepassingsgebieden en verwante onderzoeksdomeinen. Hoofdstuk 6 sluit de thesis af met een korte samenvatting van de belangrijkste resultaten en formuleert nieuwe uitdagingen voor de toekomst.

Hoofdstuk 2: Blokcijfers

In het tweede hoofdstuk van deze thesis verdiepen we ons in cryptografische systemen. Onze focus richt zich vooral op vercijferingsalgoritmes. Deze hebben tot doel om data (klaartekst) om te zetten in cijfertekst zodat enkel een persoon die de geheime sleutel kent de klaartekst opnieuw kan bekomen. In het algemeen kunnen 2 soorten cryptografische vercijferingssystemen onderscheiden worden: symmetrische en asymmetrische cryptografie. In symmetrische systemen gebruiken beide partijen dezelfde sleutel, terwijl bij asymmetrische systemen beide partijen een verschillende sleutel hanteren (een vercijferingssleutel, die publiek kan zijn, en een private ontcijferingssleutel).

Symmetrische cryptografie wordt op zijn beurt opgesplitst in blokcijfers en stroomcijfers. Stroomcijfers genereren uit de geheime sleutel een 'sleutelstroom', die bij de klaartekst opgeteld wordt. Bij het ontcijferen wordt dezelfde sleutelstroom gegenereerd. Stroomcijfers transformeren klaartekst van arbitraire lengte, terwijl blokcijfers op vaste blokken data opereren. Blokcijfers bestaan typisch uit een opeenvolging van ronden die bestaan uit eenvoudige bouwblokken, waarbij elk van deze ronden sleutelafhankelijk zijn.

Cryptanalysetechnieken

Het is nuttig om binnen de context van deze thesis de traditionele aanvallen voor te stellen, wegens een verband met aanvalstechnieken in een white-box model. Een eerste aanvalsmodel is het meest traditionele en noemen we vaak het black-box aanvalsmodel. Hierbij wordt een cryptografische implementatie als een

zwarte doos gezien en is enkel orakeltoegang mogelijk. Dit wil zeggen dat een aanvaller enkel over klaartekst en cijfertekst informatie beschikt, en op basis hiervan sleutelinformatie probeert te bekomen. Dergelijke aanvallen zoeken typisch naar statistische verbanden tussen verschillen aan de ingang en verschillen aan de uitgang (lineaire en differentiële cryptanalyse), of wiskundige vergelijkingen op te stellen voor het (sleutelgeïnstantieerde) algoritme (algebraïsche cryptanalyse).

Een tweede aanvalsmodel omvat nevenkanaalaanvallen. Dergelijke aanvallen gebruiken informatie die 'lekt' tijdens het vercijferingsproces en eigen is aan de manier waarop de cryptografische algoritmes zijn geïmplementeerd. Deze aanvallen zijn typisch uitvoerbaar wanneer de aanvaller een beperkte toegang heeft tot het uitvoerende toestel. Voorbeelden van nevenkanaal informatie zijn stroomverbruik van het toestel, elektromagnetische straling, en uitvoeringstijd. Bij actieve nevenkanaalaanvallen kan een aanvaller proberen het vercijferingsproces te beïnvloeden door bijvoorbeeld fouten te introduceren (door middel van straling), de voedingsspanning naar het toestel te onderbreken of andere parameters (bv temperatuur) te beïnvloeden.

In een white-box aanvalsmodel stellen zich alle mogelijke aanvallen die hierboven beschreven zijn. Een aanvaller heeft volledige toegang tot het uitvoerend platform en kan zo alle mogelijke parameters wijzigen. Wegens een volledige toegang tot de software implementatie kan de aanvaller ongelimiteerd black-box aanvallen uitvoeren, alsook de interne rondestructuur van een blokcijfer aan een analyse onderwerpen. We stellen software-implementaties van vercijferingsalgoritmes dus aan een erg krachtig aanvalsmodel bloot.

Hoofdstuk 3: White-box implementaties

Het bekomen van veilige software-implementaties verwijst in de eerste plaats naar de methodologie van het implementeren van bestaande cryptografische algoritmes. Wanneer een aanvaller namelijk volledige toegang heeft, is de manier van implementeren de enige verdedigingslijn. Logischerwijs dienen zich enkel vercijferingschema's aan, die al veilig zijn tegen black-box aanvallen, vermits deze ook in een white-box aanvalsmodel kunnen worden uitgevoerd. Omgekeerd zijn alle implementaties die veilig zijn tegen white-box aanvallen, inherent veilig tegen eender welke black-box en nevenkanaal aanvallen.

In 2002 deden Chow *et al.* [43, 42] een voorstel hoe dit kan worden aangepakt met voorbeeld implementaties toegepast op DES en AES. Het algemene idee is om alle operaties waaruit een blokcijfer bestaat, om te zetten naar een netwerk van sleutelafhankelijke opzoektabellen, en vervolgens deze tabellen te randomizeren door het invoegen van willekeurige, parengewijs opheffende, encoderingen. De hoop is dat door analyse van het netwerk gerandomiseerde opzoektabellen de sleutelinformatie zodanig moeilijk te extraheren valt, dat een aanvaller evengoed

een black-box aanval had kunnen uitvoeren. In dit hoofdstuk van deze thesis willen we de veiligheid van deze techniek nagaan.

Veiligheid

Het achterliggende idee dat een dergelijke techniek veilig kan zijn, komt uit het concept van *lokale veiligheid*. Beschouw een reeks van 3 opzoektabellen $L_3 \circ L_2 \circ L_1$, waarbij L_2 geheime sleutelinformatie bevat. Bijvoorbeeld $L_2(x) = f(x) \oplus k$, met f een gekende functie, en k geheime sleutel informatie (in overeenstemming met het Kerckhoffs principe). In een white-box model heeft de aanvaller toegang tot de beschrijving van L_2, en kan deze dus evalueren voor de waarde 0 om de sleutelinformatie te bekomen: $k = L_2(0)$.

Om dergelijke aanval tegen te gaan, worden de opzoektabellen als volgt geëncodeerd:

$$\begin{array}{lcl} L_1 & \rightarrow & L_1' = b_1 \circ L_1 \\ L_2 & \rightarrow & L_2' = b_2 \circ L_2 \circ b_1^{-1} \\ L_3 & \rightarrow & L_3' = L_3 \circ b_2^{-1}\,, \end{array}$$

waarbij b_1 en b_2 bijecties zijn waarvan hun dimensies overeenkomen met deze van L_2. Het resultaat is een gerandomiseerde reeks van 3 opzoektabellen $L_3' \circ L_2' \circ L_1'$, met als fundamentele eigenschap dat L_2' geen sleutelinformatie lekt. Deze fundamentele eigenschap noemen we *lokale veiligheid*. Indien een aanvaller sleutelinformatie wil bekomen, moet hij eerst informatie hebben over b_1 en b_2, door analyse van respectievelijk L_1' en L_3'. We dwingen aldus de aanvaller tot een analyse van de hele reeks.

Deze methodologie wordt uitgebreid naar blokcijfers. De bedoeling is dat we een aanvaller dwingen een analyse te doen van de hele implementatie (een netwerk opzoektabellen), teneinde sleutelinformatie te bekomen. We willen dat de moeite die hij hiervoor moet doen, ten minste equivalent is met het uitvoeren van een black-box aanval.

De veiligheid van deze implementatiestrategie valt moeilijk te omschrijven. In eerste fase beschreven Chow *et al.* metrieken zoals diversiteit en dubbelzinnigheid. De diversiteit metriek beschrijft hoeveel verschillende instanties gemaakt kunnen worden met dezelfde sleutel, dubbelzinnigheid kwantiseert hoeveel kandidaat sleutels er zijn voor eenzelfde instantie. Metrieken zijn echter slechts indicatief, en enkel van toepassing uit gebrek aan veiligheidsbewijzen of analyse technieken. In het vervolg van dit hoofdstuk wordt een analyse van de state-of-the-art white-box implementaties gepleegd. In hoofdstuk 4 gaan we op zoek naar veiligheidsbewijzen.

Analyse

Onze analyse richt zich initieel op de white-box implementaties van DES en AES. We onderscheiden twee soorten aanvallen: differentiële aanvallen en algebraïsche aanvallen. In deze verhandeling lichten we beide aanvalstechnieken toe en geven we een beschrijvend overzicht van enkele voorgestelde praktische white-box aanvallen.

Differentiële aanvallen bestuderen de propagatie van het verschil tussen klaarteksten. Vermits in een white-box omgeving alle operaties zichtbaar zijn, kunnen we tijdens de uitvoering wijzigingen doorvoeren (fouten introduceren), en bestuderen hoe deze zich doorheen de implementatie propageren. Bestaande differentiële aanvallen op white-box implementaties richten zich op het introduceren van fouten in de eerste of laatste ronde, waardoor ze sterk afhankelijk zijn van ingang-uitgang encoderingen.

In ons onderzoek stellen we een nieuwe differentiële aanval voor, die op de interne rondestructuur kan worden uitgevoerd, en daardoor onafhankelijk is van externe encoderingen. Het succes van deze aanval berust op het feit dat het moeilijk blijkt om differentiële eigenschappen van DES te verbergen in de white-box implementatie. Ondanks het feit dat deze aanval specifiek is voor DES, kan een analoge aanpak ook op andere implementaties toegepast worden, in het bijzonder op Feistel blokcijfers.

Algebraïsche aanvallen omschrijven een cryptografisch primitief door middel van algebraïsche vergelijkingen, om deze vervolgens op te lossen. Dit werd in de context van white-box cryptografie voor het eerst voorgesteld op AES implementaties, waarbij de analyse van een geobfusceerde ronde de encoderingen van de opzoektabellen grotendeels ongedaan kan maken. Het blijkt dat AES enkele eigenschappen heeft die zo'n aanval mogelijk maken. Verder onderzoek heeft dergelijke aanvallen uitgebreid naar een grotere klasse vercijferingsalgoritmes.

Omdat blijkt dat bepaalde bewerkingen leiden tot een algebraïsche aanval, voeren we een analyse uit op de bouwblokken van blokcijfers. We presenteren meer bepaald een aanval op de optellingsoperatie. Dit basisblok blijkt inherent onveilig te zijn tegen aanvallen die de encoderingen proberen ongedaan te maken, wat een impact heeft op de white-box implementaties van AES en DES.

Dit geeft een bijzonder interessant resultaat: ondanks het feit dat white-box cryptografie initieel enkel bestudeert *hoe* een cryptografische veilig kan worden geïmplementeerd (gegeven het primitief), blijkt het ook eisen te stellen op het ontwerp van dit primitief. Als resultaat van dit hoofdstuk concluderen we dan ook met enkele ontwerpcriteria voor het ontwikkelen van nieuwe primitieven (blokcijfers) die potentieel veilig kunnen worden geïmplementeerd.

Alle beschreven aanvallen hebben tot doel sleutelinformatie uit de implementaties te bekomen. Het veilig implementeren omvat echter niet enkel als doelstel-

ling het geheimhouden van sleutelinformatie, maar ook het behouden van andere eigenschappen van cryptografische primitieven. Bijvoorbeeld de eigenschap dat vercijferingsalgoritmes slechts in één richting uitgevoerd kunnen worden (enkel vercijfering, geen ontcijfering mogelijk). Alvorens dit concept in hoofdstuk 4 formeel aan te pakken, tonen we hoe uit de white-box implementaties van de AES en DES encryptieschema's, hun corresponderend ontcijferingsalgoritme kan worden afgeleid.

Hoofdstuk 4: Theoretisch model

In het vorige hoofdstuk werden praktische white-box implementaties voorgesteld. In de literatuur werden echter geen formele definities voor white-box cryptografie beschreven, alsook geen bewijzen van veiligheid. Gezien de succesvolle cryptanalyse van de bestaande white-box implementaties blijft het een open vraag of veilige white-box implementaties al dan niet bestaan.

White-box cryptografie heeft als doel om een niveau van robuustheid te bereiken voor cryptografische software implementaties die onderhevig zijn aan aanvallen vanuit het uitvoeringsplatform. Dit stemt deels overeen met de technieken van *code obfuscatie*, waarbij het doel is een programma P zo te implementeren dat bepaalde karakteristieken van dit programma verborgen worden. Verschillende theoretische modellen voor code obfuscatie zijn reeds voorgesteld. Het is echter niet duidelijk of deze het begrip white-box cryptografie kunnen omvatten.

In dit hoofdstuk maken we een vergelijk tussen white-box cryptografie en obfuscatie, en stellen dat de huidige definities van code obfuscatie niet volstaan voor WBC. We stellen we een eerste model voor WBC op en formaliseren het concept white-box cryptografie door middel van een *white-box eigenschap*. Dit maakt een parallel met traditioneel onderzoek in theoretische modellen en bewijsbare veiligheid in de cryptografie. In het tweede deel van dit hoofdstuk onderzoeken we de eigenschappen van white-box cryptografie met ons nieuwe model. We stellen enkele positieve en negatieve resultaten voor. Zo tonen we aan dat een white-box implementatie van een cryptografisch primitief onmogelijk alle eigenschappen die het in een black-box aanvalsmodel heeft, kan behalen in een white-box aanvalsmodel. Als positief resultaat tonen we aan dat er cryptografische primitieven bestaan die aan zinvolle eigenschappen kunnen voldoen in een white-box omgeving. Zo stellen we een concreet symmetrisch encryptieschema voor dat semantisch veilig is, en zodoende kan worden geïmplementeerd dat het ook in een white-box aanvalsmodel semantisch veilig blijft.

Code obfuscatie

Het onderzoek naar formele modellen voor code obfuscatie is een actief onderzoeksdomein. Aansluitend geven we een kort overzicht van de formele definities, en omvatten de belangrijkste resultaten.

We noemen een *obfuscator* het proces dat een algoritme veilig implementeert. Over het algemeen dient een obfuscator $\mathcal{O}$ aan de volgende drie eigenschappen te voldoen:

- Functionaliteit – Het geobfusceerde programma $\mathcal{O}(P)$ dient functioneel equivalent te zijn aan het originele programma P.
- Equivalente grootte en performantie – Het geobfusceerde programma $\mathcal{O}(P)$ dient realistisch te blijven in verhouding met het originele programma.
- Virtuele Black-Box Eigenschap

Vooral de laatste eigenschap is van belang en omvat de veiligheid van een obfuscator. Het is ook net in deze eigenschap dat de verschillende modellen voor code obfuscatie zich onderscheiden. In de predicaat gebaseerde definitie krijgt de aanvaller A het geobfusceerde programma $\mathcal{O}(P)$, en maakt die een uitspraak (π) over het programma P. Indien een simulator S bestaat, met enkel toegang tot de functionaliteit van P (= orakel toegang, of invoer-uitvoer toegang), die met een grote waarschijnlijkheid dezelfde uitspraak $\pi(P)$ kan bekomen, dan heeft de aanvaller A geen voordeel gehad aan white-box toegang. Deze situatie is afgebeeld in Figuur 4.1. Een obfuscator $\mathcal{O}$ voldoet aan de predicaat gebaseerde virtuele black-box eigenschap, indien voor alle mogelijke programma's P en predicaten π, er voor elke aanvaller A steeds een simulator S kan worden opgesteld, tot op een verwaarloosbaar aantal gevallen na. Een andere definitie is de IND-definitie, waarbij een statisch vergelijk wordt gemaakt tussen de uitvoer van de aanvaller A, en een simulator S.

Het probleem is dat er geen consensus is over welk model geschikt is voor obfuscatie. De verschillende modellen hebben elk hun voor- en nadelen. Zo zijn onder de IND-definitie slechts een beperkt aantal deterministische programma's obfusceerbaar. Cryptografisch nuttige programma's, zoals deterministische vercijferingsalgoritmes, kunnen dus onder dergelijke strikte vereisten niet worden geobfusceerd. Soepelere definities, zoals de predicaat gebaseerde, zijn dan weer te zwak om een zinvol resultaat te bekomen. Vermits predicaten gedefinieerd zijn over het originele programma, kan deze definitie het lekken van non-black-box informatie niet omvatten.

Bovendien lijkt het moeilijk om cryptografische vereisten, zoals bv. de niet-inverteerbaarheid van vercijferingsalgoritmes, te omvatten door middel van obfuscatie definities. We willen deze problemen aanpakken door een nieuw model voor te stellen dat specifiek is voor white-box cryptografie.

White-box cryptografie

Aan de basis van ons theoretisch model staan *security noties*. Hierbij volgen we de strategie van formele modellen in traditionele cryptografie en bewijsbare veiligheid. Een security notie is een formele beschrijving van de capaciteiten van de aanvaller, en wat beschouwd wordt als een succesvolle aanval. Dit wordt in het algemeen omschreven als een interactie tussen een uitdager en een aanvaller, wat we een experiment noemen.

In het voorgestelde model maken we een vergelijking tussen twee experimenten: het black-box experiment, waarbij een aanvaller enkele orakeltoegang heeft tot functies Q, en een white-box experiment, waarbij een aanvaller naast orakeltoegang tot de functies Q ook de software-implementatie $\mathcal{O}(Q_i)$ verkrijgt. Dit komt intuïtief overeen met de strategie in bewijsbare security, waarbij een vergelijking of reductie wordt gemaakt tussen een cryptografisch primitief en zijn geïdealiseerde tegenhanger, of met de aanpak in code obfuscatie waarbij een vergelijking wordt gemaakt tussen een aanvaller en een simulator. We stellen dat een obfuscator $\mathcal{O}$ voldoet aan de white-box eigenschap voor een bepaalde security notie sn en familie Q, wanneer de beste aanvaller in het white-box experiment niet significant meer informatie kan bekomen (binnen het kader van de security notie), dan de beste aanvaller in het black-box experiment.

Een belangrijke vaststelling die we maken is *obfusceerbaarheid*. Een security notie omschrijft de capaciteiten van een aanvaller. Sommige security noties beperken een aanvaller in de operaties die kunnen uitgevoerd worden op een bepaald primitief. Zo staat de IND-CCA2 security notie (zie algoritme 6) niet toe dat een decryptieoperatie wordt uitgevoerd op een bepaalde cijfertekst. Echter, wanneer de decryptieroutine wordt geobfusceerd en ter beschikking wordt gesteld van de aanvaller, kan een dergelijke beperking niet ingevoerd worden. We kunnen dus stellen dat de decryptieoperatie niet obfusceerbaar is wanneer van het encryptie algoritme IND-CCA2 security verwacht wordt.

Resultaten

Dit model geeft ons de mogelijkheid om de veiligheid van white-box implementaties te onderzoeken. Aan de hand van enkele voorbeelden tonen we aan dat er een wezenlijk verschil is tussen white-box cryptografie en obfuscatie. De white-box eigenschap is namelijk gedefinieerd voor een familie functies met geassocieerde security notie, terwijl obfuscatie gedefinieerd is onafhankelijk van een security notie.

Een belangrijke vraag die we stellen, is of er een familie niet-triviale functies (zoals bv. vercijferingsalgoritmen) bestaat, waarvoor een obfuscator kan worden opgesteld die zorgt dat alle security noties waaraan de familie voldoet in black-box, ook voldaan zijn in een white-box model. We tonen aan dat dit niet het geval

is, namelijk dat er een security notie bestaat waaraan niet voldaan kan worden in white-box ondanks het feit dat in black-box wel aan deze notie voldaan is.

Vermits de security notie die hiervoor gebruikt wordt, slechts een beperkt praktisch nut heeft, kunnen we ons het volgende afvragen: bestaat er een zinvolle security notie voor een zinvolle familie niet-triviale functies waaraan ook de white-box implementatie voldoet? We tonen aan dat dit, onder aanvaardbare veronderstellingen, het geval is. Meer bepaald beschrijven we concreet een symmetrisch vercijferingsalgoritme dat IND-CPA veilig is, waarvan zijn white-box implementatie een asymmetrisch vercijferingsalgoritme is dat ook voldoet aan de IND-CPA security notie.

Tenslotte zetten we ook een uitbreiding van ons model naar probabilistische algoritmes uiteen, vermits deze in de cryptografie van praktisch groot belang zijn. Dit hoofdstuk kan gezien worden als een eerste stap om de fundamenten van white-box cryptografie op een gelijk niveau te brengen als wat bereikt is in de formele studie van code obfuscatie.

Hoofdstuk 5: Toepassingen

Het onderzoek in white-box cryptografie richt zich op het veilig implementeren van cryptografische primitieven in software. Bijgevolg leunt dit onderzoek dicht aan bij praktische toepassingen. In dit hoofdstuk stellen we 4 verwante toepassingsdomeinen voor, waarop white-box cryptografie van nut kan zijn (en vice versa). We stellen kort de 4 geïdentificeerde domeinen voor.

Nieuwe cryptografische primitieven

Een eerste toepassing van white-box cryptografie is de eigenschap om nieuwe cryptografische primitieven te maken. White-box cryptografie kan symmetrische vercijferingsschema's namelijk omzetten in asymmetrische vercijferingsschema's. Bij symmetrische schema's wordt eenzelfde geheime sleutel gebruikt voor zowel vercijfering als ontcijfering. Bij asymmetrische schema's is er een publieke vercijferingssleutel, en een private ontcijferingssleutel. Wanneer we een symmetrisch vercijferingsalgoritme samen met de symmetrische sleutel obfusceren, kan het resultaat als publieke sleutel worden gebruikt. Iedereen kan vercijferen met de bekomen implementatie, terwijl enkel met kennis van de symmetrische sleutel decryptie mogelijk is.

De bekomen systemen zijn echter pas veilig wanneer deze voldoen aan de CPA notie. Dit geeft meteen het belang aan van onze studie in hoofdstuk 4. In hoofdstuk 3 hebben we ook aangetoond (door middel van een praktische aanval) dat de voorgestelde white-box implementaties van AES en DES hieraan niet voldoen.

Hardware

Vermits white-box cryptografie implementaties beschermt tegen software aanvallen, beschermt deze ook tegen zwakkere aanvalsmodellen, zoals bijvoorbeeld nevenkanaal aanvallen. We stellen deze techniek dan ook voor om veilige implementaties op Gsm's of TV set-top boxes te ontwikkelen.

Software geeft enkele inherente voordelen ten opzichte van hardware oplossingen. Ze zijn goedkoper om te produceren (geen apparatuurkosten), en veel flexibeler. Zo kan software bijvoorbeeld via het Internet worden ge-update. Het is dus interessant om te zien welke functionaliteiten door middel white-box cryptografie kunnen worden beschermd, in plaats van te berusten op hardware-beveiliging.

Berekeningen in het vercijferde domein

Homomorfe functies zijn functies die rechtstreeks op geëncodeerde data berekeningen kunnen uitvoeren. Zo kan bijvoorbeeld een optelling van x_1 met x_2 ook rechtstreeks uitgevoerd worden op hun encodering $E(x_1), E(x_2)$, indien het vercijferingsschema homomorfe eigenschappen heeft. Dit heeft interessante toepassingen om de privacy van gevoelige gegevens te garanderen. Het obfusceren van een cryptografisch schema dat 2 input waarden ontcijfert, een bewerking op uitvoert, en terug vercijfert, kan gezien worden als een homomorfe functie.

Een ander verwant onderzoeksgebied is dat van 'multi-party' evaluatie. In deze opstelling wensen een aantal partijen met private invoer een gemeenschappelijke functie f te evalueren met hun invoer. Dit zonder hun private invoer aan de andere partijen vrij te geven, en een soort garantie te bekomen dat de evaluatie correct is verlopen. De technieken om dit te realiseren, zijn gebaseerd op 'garbled circuits'. Dit zijn een soort gerandomiseerde implementaties, wat sterk overeenkomt met de aanpak in white-box cryptografie.

Softwarebeveiliging

Dit onderzoek heeft als doel om software-implementaties van cryptografische primitieven te beschermen tegen white-box aanvallen, en vormt daarmee een onderdeel van algemene software beschermingstechnieken. De doelstellingen van softwarebeveiliging zijn in het algemeen het geheim houden van operaties of data (obfuscatie); zorgen dat de functionaliteit niet gewijzigd kan worden; en garanderen dat software correct uitgevoerd wordt.

Tijdens het onderzoek op white-box cryptografie, werden diverse technieken ontwikkeld die van toepassing zijn op softwarebeveiliging.

Hoofdstuk 6: Besluit en verder onderzoek

White-box cryptografie heeft als doelstelling software-implementaties van cryptografische primitieven te beschermen tegen allerhande aanvallen. In deze thesis hebben we de verschillende praktische white-box implementaties bestudeerd. De cryptanalyses hiervan leiden tot nieuwe inzichten en een strategie voor het ontwikkelen van nieuwe blokcijfers die geschikt zijn om in software te gebruiken.

In een tweede fase van het onderzoek stellen we een theoretisch raamwerk van white-box cryptografie voor. Dit laat ons toe om verschillende eigenschappen voor WBC te onderzoeken. Wat wel of niet binnen de mogelijkheden ligt. We stellen een symmetrisch vercijferingsschema voor, die bewijsbaar veilig geïmplementeerd kan worden. Verder onderzoek dient te leiden tot een evaluatie van andere klasses functies, zoals probabilistische vercijferingsschema's.

Het onderzoek wordt beëindigd met een overzicht van toepassingen waarvoor WBC van nut kan zijn, en verwante onderzoeksgebieden. Dit omvat onder andere het ontwerp van nieuwe cryptografische algoritmes, berekeningen uitvoeren rechtstreeks op vercijferde data, het gebruik van hardware, en software beveiliging. Binnen elk van deze domeinen geven we aanzet hoe WBC gebruikt kan worden.

Chapter 1

Introduction

We live in an information society. Increasingly, we rely on the exchange and processing of information. Evidence of this evolution is the rapid growth of communication networks, and a trend towards complex software applications with strong security requirements. Examples of these are the increasing use of portable devices and wireless networks; communication with friends and colleagues via e-mail and chat; the launch of (interactive) digital television and other media platforms (e.g., iTunes); on-line banking and purchase of goods and services; online-gaming; GPS navigation; professional and social networks (e.g., LinkedIn and Facebook); and many more. In one way or another, these new trends affect our daily activities in many ways, at home and in our professional life. On the downside however, we become increasingly dependent on the information infrastructure that empower our information society, and hence potentially vulnerable to attacks on them. In recent years, this has been illustrated by attacks on Internet servers, credit card fraud, hacking of banking applications and on-line games, cell phones and TV set-top boxes, phishing, privacy violation, botnet threats, and so forth.

In order to support our information society for the next years, and take advantage of the opportunities that it enables, the need for trustworthy information infrastructure is growing. The trend towards complex software applications with strong security requirements, increasingly demands for qualitative protection technologies. One prominent building block to enable information security is *cryptology*.

1.1 Cryptology

The word cryptology is derived from the Greek words *kryptós*, meaning 'hidden', and *logos*, meaning 'word'. Strictly speaking, it is the science that studies how to hide confidential information. Cryptology comprises of two complementary fields. *Cryptography* is the study and practice of hiding information, while *cryptanalysis* is the study of methods to obtain knowledge from hidden information.

The foundations of cryptography originate from Shannon, who is regarded as the founder of information theory. In his seminal work on a mathematical model for cryptography in 1948 [173], he described the basic model for a cryptosystem. This typical scenario of cryptography, depicted in Fig. 1.1, consists of two parties (traditionally denoted as Alice and Bob) who wish to exchange confidential information.

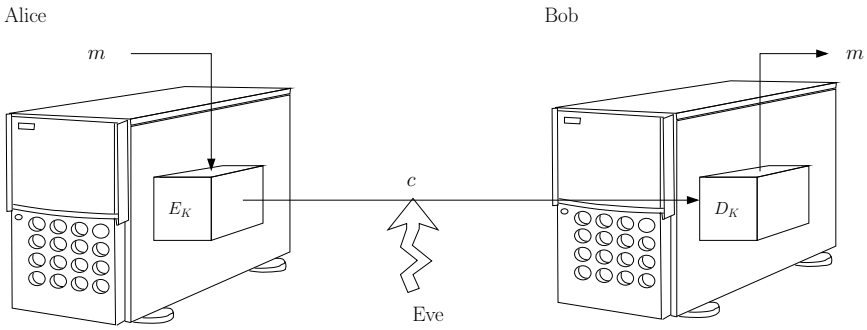

Figure 1.1. A cryptosystem

In this classical model, Alice and Bob want to transmit confidential messages m over an insecure channel in such a way that an adversary (Eve) eavesdropping on the channel is not able to learn anything about the message. In modern cryptography, Kerckhoffs' principle states that only a secret key k is unknown by the adversary, while the encryption and decryption algorithm are known by all parties. This secret key is a priori exchanged between Alice and Bob.

Instead of the *plaintext* message m, Alice will send an encrypted *ciphertext* message c to Bob over the insecure channel. The ciphertext is computed by Alice using the encryption algorithm E, instantiated with the secret key k: $c = E_k(m)$, which Bob is able to decrypt using the decryption algorithm D. The same key will be used for decryption, such that $D_k(c) = D_k(E_k(m)) = m$.

We will denote this basic model as the *black-box model*, because Eve observes only the channel between the communicating parties. In Sect. 3.1, we elaborate on the different attack models.

In [157], Rivest summarized cryptography as follows: "Cryptography is about communication in the presence of adversaries". In this thesis, the exact interpretation of this sentence is of great importance. In particular, "presence of adversaries" can be interpreted in various ways. The problem statement addressed in this thesis is actually to find out how cryptography could be deployed in the presence of the most powerful adversaries.

Although cryptography has received much attention in the last decades, the science of "secret writing" is in fact far older. We give an overview of some historic facts in the development of cryptography, to motivate the research carried out in this thesis.

1.1.1 The Caesar Cipher

One early occurrence of cryptography is a cipher called "Caesar Cipher" (after Julius Caesar), which was used by him and his successors in the times of the Roman Empire to send directives from Rome to the legions and governors [121]. In this cipher, every letter is replaced by its third successor in the alphabet (A $\rightarrow$ **D**, B $\rightarrow$ **E**, ...).

plaintext	A	M	E	S	S	A	G	E
ciphertext	**D**	**P**	**H**	**V**	**V**	**D**	**J**	**H**

However, as soon as one knows that a ciphertext is only a translation of the plaintext, it is fairly easy to break the Caesar cipher. Since there are only 26 possible translations (in the Latin alphabet), an adversary can compute all possible plaintexts for a given ciphertext, until something readable appears. Nevertheless, the cipher worked sufficiently well in Roman times.

Later, ciphers were improved with 'transposition vectors'. Instead of translating every symbol with a single fixed number of positions, more complex translation vectors were used. Consider the following example of the Vigenère cipher [103, page 146] below, where the translation vector is `KEY`. The *key* word is repeated to have the same length as the plaintext, and the ciphertext is the addition of the plaintext with the expanded key ($A + K \rightarrow$ **L**; $M + E \rightarrow$ **R**; ...). The Vigenère cipher with key word 'C', corresponds to the Caesar cipher.

plaintext	A	M	E	S	S	A	G	E
transposition	K	E	Y	K	E	Y	K	E
ciphertext	**L**	**R**	**D**	**F**	**X**	**Z**	**R**	**J**

Most ciphers in the history of cryptography [103], like the Vigenère cipher, are *polyalphabetic* ciphers. These are based on substitution using multiple substitution alphabets that are derived from a key word. Even the Enigma Machine (most famous because of its use by the German military in World War II) is fundamentally a polyalphabetic cipher. However, these ciphers are inherently vulnerable to frequency analysis [3], where patterns are searched for in the ciphertexts. This inspired research towards modern cryptography.

1.1.2 Kerckhoffs' Principle

In the past, cryptography was mainly used for military applications and diplomatic communications. For this purpose, governments developed cryptographic ciphers, and kept their designs confidential. In fact, those ciphers were only secure because of this confidentiality. As soon as the design was leaked or reverse engineered, enemies could break the ciphers and start eavesdropping and forging messages.

In 1883, August Kerckhoffs [104] defined the *Kerckhoffs' principle*, that states that a *cryptosystem*[1] should be secure even if everything about the system, except a *key*, is public knowledge. This principle is widely accepted, because it has been shown in numerous cases that *security through obscurity* (keeping the design confidential) is bad practice. The adoption of this principle is fundamental for modern research in cryptography, both for government research institutes (since World War II) and universities and industry (since around 1980).

Where in the past, the security of a cipher was only verified by a few experts, they are now subject to public scrutiny because of this principle. This led to a public debate and the search for secure cryptographic ciphers.

1.1.3 One-Time Pad

A year after he founded information theory, Shannon published a paper that proved that unbreakable cryptography was possible [173]. (He did this work prior to 1945, but at that time it was classified.) The scheme is called the one-time pad or the Vernam cipher, after Gilbert Vernam, who had invented it near the end of World War I.

plaintext	A	M	E	S	S	A	G	E
key	R	A	N	D	O	M	K	E
ciphertext	**S**	**N**	**S**	**W**	**H**	**N**	**R**	**J**

[1] The term *cryptosystem* is used as the short writing for *cryptographic system*. That is, any system which involves cryptography, and hence contains a *cryptographic primitive* as a component. A formal definition is presented in Chapter 2.

The One-Time Pad offers *unconditional security against unbounded adversaries*. This means that, even with unlimited amount of computing power, an adversary cannot compute any information of the secret key or the plaintext. This is often denoted as *perfect* security [189, page 8]. Up to now, only few ciphers provide this level of security. Unfortunately, it is not practical because keys are as long as the messages that need to be encrypted, and cannot be re-used. Therefore, it is used only very rarely, e.g., it was used to secure the communication channel between the United States White House and the Russian Kremlin [103, page 715].

1.1.4 Modern Cryptography

Since the early 1970s, cryptography has broadened its scope. Where in the past, cryptography was exclusively about securing messages against eavesdropping (confidentiality of information), modern cryptography also covers problems such as message integrity, authentication, and non-repudiation. We refer to the *Handbook on Applied Cryptography* by Menezes, Van Oorschot, and Vanstone [129] for an excellent treatment of these issues.

Secondly, modern cryptography differs from 'classical' (pre-1977) cryptography in its methods to assess the security of cryptosystems. In the classical approach, only designers and experts justified the security of a cryptosystems by 'failure to break'. As long as there was no break (in spite of usage), a system was believed to be 'secure'. Breaking a system can have a variety of meanings: obtaining secret key information or plaintext information; forging digital signatures; corruption of authenticated messages, and so forth. In modern cryptography, we can distinguish three ways to assess the security of a cryptosystem:

1. *Direct proof of security.* Show that a cryptosystem is unconditionally secure, based on information theoretical proofs. Unfortunately, only few cryptosystems are shown to be information theoretically secure, and are largely impractical.

2. *Proof by reduction.* Prove the security of a cryptosystem by reduction to a hard mathematical problem. I.e., when an adversary would be able to break the cryptosystem, the mathematical problem would be easy to solve. These mathematical problems are typically NP hard[2] problems, which even the best mathematicians do not seem to be able to solve for many years (and hence we accept the hardness assumption). Often, (sub-)exponential problems are accepted also. For example, the best known

[2] A problem is *NP (nondeterministic polynomial time)* if it is solvable in polynomial time by a nondeterministic Turing machine (see Definition 11, Chapter 4). A problem is said to be *NP-hard*, if it is at least as hard as the hardest problem in NP.

algorithm for factoring the product of two large primes, is in the general case sub-exponential, and used as a hardness assumption in many practical schemes.

3. *Failure to cryptanalyze.* Assess the security by the development of cryptanalysis techniques (security based on scrutiny).

The modern approach to evaluate the security is an open process. An open competition of experts and 'less'-experts from academics and industry, organized in challenges, conferences, publications, prizes, and so forth, where reputation is often important.

Finally, modern cryptography deals with a large variety of applications, and is no longer exclusively deployed for military or diplomatic communications. It has become a tool for a large section of our economy (both for industry as for home consumers). The vast growth of (digital) communication between all sorts of parties (supported by the explosive growth of the Internet, wireless networks, and cell phones), and a trend towards complex software applications that demand stronger security requirements (e.g., on-line banking, media platforms, and games), enforced the interest in cryptology in all its aspects.

Moreover, while early systems worked well in military or diplomatic applications, where a fixed hierarchy of people were authorized to have access to and knowledge of the deployed system, modern systems work in a completely different setting. This gives a new dimension to Rivest's description of cryptography, where adversaries could have completely different capabilities.

1.2 Motivation

Commonly deployed ciphers are designed to operate in the *standard model* as depicted in Fig. 1.1. In this a model, it is assumed that the communication end points and computing environments are trusted. That is, it is assumed that the cipher execution (encryption/decryption, instantiated with a secret key) cannot be observed or tampered with. Only its functionality is accessible by an adversary, hence it is often denoted as the *Black-Box Model*.[3]

However, the assumptions made in the past may often not be applicable in current technology. In the past decade, the applications for which cryptographic techniques were deployed have changed dramatically. As a result, one can no longer assume that the communication end points are trusted entities. This has a huge impact on the security of cryptographic implementations. When such an

[3] In the literature, one may encounter cryptographic models, such as the *bare model*, *plain model*, or *(random) oracle model*. These are examples of a *black-box model*.

implementation resides in a hostile environment, an adversary may be able to observe and tamper with the implementation to extract information about the cryptographic key. As a result, methods that were developed in the past to assess the security of ciphers, may no longer suffice for many modern applications.

Below, we present two examples where this conventional black-box model of cryptography fails. In Chapter 5, we elaborate in more detail on these and more applications where white-box cryptography could be deployed.

1.2.1 Mobile Agents

Mobile agents [163, 162, 94, 127, 86] are programs, which may be sent off from a client computer to a remote server for execution. Often, they are even able to travel around in a (public) network. Their goal is to fulfill the task that was given to them by their owner, without any interaction with the owner during the execution of that task. They have been proposed as a method for performing transactions and information retrieval in networks. Typical examples of mobile agent systems would be flight ticket ordering systems, or online auctioning.

In the case of the flight ticket ordering system, a mobile agent is sent out by its owner, who wants to find the cheapest flight from A to B. This agent will visit different travel agencies' and flight companies' websites in order to find the cheapest ticket, and proceed with its purchase. The whole process should run without any interaction with the owner who eventually wants to obtain the electronic flight ticket.

Clearly, there are a few unconventional threats related to such a system. First of all, the servers of travel agencies and flight companies might not behave as trusted end points. It is in their interest to attack these mobile agents. E.g., rewrite the code from 'find cheapest flight' to 'find my € 200 flight', or force the agent to buy a ticket on their servers. Secondly, in order to enable the purchase of a ticket, the agent might need to be able to sign a contract or perform a payment. Therefore, a public key cryptosystem needs to be deployed, such that the mobile agent can create a digital signature for an electronic ticket. However, because the mobile agent should not interact with the owner in the process of purchase, the private signing key needs to be stored in the code of the mobile agent. It is in the interest of the (malicious) servers, to obtain this private key information, such that they can sign arbitrary electronic documents, and hence purchase any goods in name of the author.

Hence a natural question arises: can we embed confidential data inside software, albeit that the execution platform is untrusted? This is a question that *white-box cryptography* attempts to address.

1.2.2 Digital Rights Management

A topic that has been at the center of the public debate, is that of Digital Rights Management (DRM). Generally speaking, this topic covers a substantial set of applications where the common goal is to restrict access to and consumption of content, often based on top of a role-based access control system.

For example in the pay TV scenario, a broadcasting company would like to distribute their multimedia content (movies) on an existing (public) network, and restrict access such that only valid subscribers can watch the movie. These subscribers should not be able to copy the movie, nor should non-subscribers be able to watch a movie. This requires the content to be sent over the public network in a garbled (encrypted) form. Figure 1.2 depicts a simplified view of how such an architecture could look like.

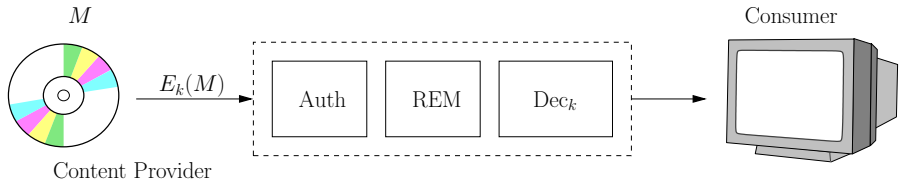

Figure 1.2. Basic DRM architecture

The multimedia content M is sent in encrypted form, where E is the encryption function, k the (secret) encryption key, and *Lic* a license that is sent along. This license contains an expression of the rights (of the subscriber) on the content. I.e., a list of which access privileges users/groups have. At the side of the consumer, an application processes the license information (by means of a *Rights Expression Manager*), parses the user authentication (through the *Auth* component), and decrypts the content using the corresponding decryption routine D, instantiated with the key k. Such an application can be implemented in hardware (e.g., in a set-top box, typical for pay TV systems), or in software (e.g., iTunes [97] or WM-DRM [136]) on the client's PC. In both cases, the application is running on a platform, controlled by an entity that might be untrustworthy.

This whole infrastructure breaks down when the secret key k is compromised. In that case, an adversary could decrypt the content (due to the Kerckhoffs' principle, D is publicly known), without prior authentication and rights verification. He could then re-distribute the content without any protection, or distribute the decryption key.

Again, the question arises, whether it is possible to prevent the extraction of key information, even when the execution platform behaves maliciously.

Two recent examples of successful memory-based key extraction attacks are the AACS/BackupHD-DVD hack that lifts the AACS[4] keys from memory to enable the BackupHD-DVD tool to copy the disc [66], and the FairUse4WM utility that removes the DRM protection from Windows Media content [63].

1.3 White-Box Model

Cryptographic ciphers are commonly designed in the standard cryptographic model, denoted as the *black-box model*, where the communication end points and computing environments are trusted. As shown above, there exist applications where adversaries do not comply with this model, hence a new model needs to be formulated. We define the *white-box model* as the worst-case attack model, in which adversaries have full access to the implementation of cryptographic primitives, and complete power over their execution environment. In Sect. 3.1, we elaborate on the details of this model.

We present two examples of techniques that are available to an adversary, and are typical to the *white-box attack context*. We also suggest how these attacks can be prevented, to be able to capture the main idea of white-box cryptography.

1.3.1 Entropy Attack

When two parties wish to communicate with each other, a cryptographic key needs to be agreed upon. In the symmetric case, this key should be a priori known by the two parties, and difficult to guess by an adversary. Hence, the *secret key* should be taken at random from the set of possible keys. A metric to measure *randomness* of data, is *entropy* [173]. We say something is random when it has high entropy,[5] and in order to be difficult to guess, a key should have high entropy.

The implementation of the encryption and decryption algorithm on the other hand has low entropy, because it is a compiled executable comprising of (a limited number of) instructions. In Fig. 1.3, a binary representation of an implementation with embedded secret key is depicted. A 0-bit is represented by a black dot, a 1-bit by a white dot.

[4] The *Advanced Access Content System* (AACS) [113] is a standard for content distribution and digital rights management, intended to restrict access to and copying of the next generation of optical discs and DVDs.

[5] There exist several functions H that express the entropy of a distribution X. In general, the entropy of *certainty* is zero, while the entropy of *random* is reasonably higher. E.g., the min-entropy of the uniform distribution of n elements is n ($H_\infty(U_n) = n$).

Figure 1.3. Graphical representation of a program binary, or memory dump [172]

We use this graphical representation to illustrate how easy secret keys can be identified in binary implementations. Parts of the binary with low entropy (compiled code, function names, ...) typically show some structure,[6] as one can see on the left and right of the figure. Parts of the binary with high entropy (secret keys) look rather noisy. Hence, we can assume with a high level of certainty that the embedded secret key is located in the middle of Fig. 1.3. The exact location can be determined by more detailed examination.

This attack was presented by Shamir and Van Someren [172] in 1998. It can be applied to any data container an adversary has read access to: program binaries on hard disks, computer memory, and so forth. Unfortunately, such types of attacks are still common practice, because often the capabilities of an adversary are underestimated. This has recently been shown once again, in the case of *cold reboot* attacks on hard disk encryption keys of laptops, by Halderman *et al.* [85]. They showed that the memory remanence can be increased dramatically with simple techniques. Hence a quick reboot (of a locked computer) with a live CD, or quickly porting the memory to another machine, facilitated to copy the memory, and search for keys in it.

Defenses against an entropy attack

In the past decade, many solutions have been proposed that could prevent attacks such as the entropy attack. Most of them are *software obfuscation* techniques, designed to protect data structures against software analysis tools. Software obfuscation refers to the set of techniques that protect code against *static* and *dynamic* analysis. They make it hard for an adversary to *understand* the implementation of a program, though leaving its functionality unharmed. The obfuscated implementation of a program P is often denoted as $\mathcal{O}(P)$.

The most common approach, mainly developed in the nineties, is to unstructure data structures and break the abstraction of a program variable. Several

[6] The 32-bit Intel architecture instruction set (code IA32) has a large set of instructions, and a *reasonable entropy*, however, significantly less than cryptographic keys have.

techniques were presented by Collberg *et al.* [47], such as *variable splitting* and *transformation*. Key information could be chopped into several parts, each of them stored at different address locations in the binary, such that the original value cannot be read out easily by a static analysis tool. Or the chops of key information could be consumed by the program in such a way that they are not all stored in the memory at the same time. However, through dynamic analysis of the implementation (i.e., analysis at execution time), the memory locations can be traced, and thereby revealing the original key (see Yamauchi *et al.* [195]). Improved techniques involve linear transformations. Instead of computing with the original key value, a transformed value can be computed with. This involves a modification of the code, hence the transformations are mostly kept simple.

Software obfuscation is an active field of research, and many other techniques have been proposed to protect software code and embedded data structures. However, no technique has been presented that is able to obfuscate cryptographic primitives such that a sufficient level of confidentiality of secret key information is obtained. As a result of these efforts, by the end of the nineties, it was believed to be impossible to hide computational information into software binaries. That is, information that is used at execution time (in contrast to *passive* information, such as watermarks).

1.3.2 Key Whitening Attack

Hiding key information in software implementations of ciphers that use *key whitening* seems even more difficult. In this example we focus on block ciphers. They are a popular cryptographic primitive for concealing information, that work on fixed size input and are initiated by a secret key [129, Chapter 7]. They consists of several rounds, where for each round, a round key is generated from the secret key. Common practice in the design of block ciphers, is to add a *key whitening* operation at the end, which is the addition of an extra round key as the final operation of the encryption.

In [105], Kerins and Kursawe presented an easy way to mount an attack on software implementations of block ciphers, that have a key whitening and static substitution boxes. Figure 1.4 depicts the final operations of such a block cipher, where S denotes the substitution box (S-box) that operates on the input x, P a permutation that operates on the output of S. This is followed by a final key whitening addition with round key k_w. The output y equals to $P(S(x)) \oplus k_w$.

Because of the Kerckhoffs' principle (Sect. 1.1.2), the definition of the static S-boxes is in general public knowledge. Hence, in a white-box attack context, an adversary can find the locations of these S-boxes in the software binary, using static analysis tools such as IDA Pro [90]. These S-boxes are implemented as lookup tables, and can easily be overwritten in the software binary. When an

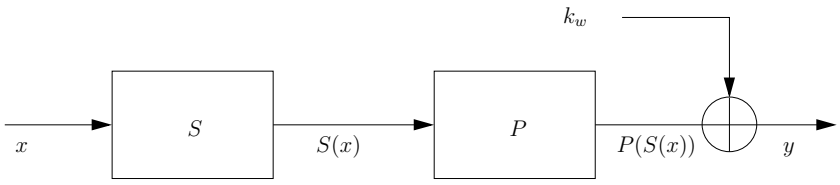

Figure 1.4. Final operations of an SPN block cipher with key whitening

adversary overwrites the lookup table with all zero's, the execution of the tampered binary will always result into the output k_w, because $P(S(x))$ equals 0 for any given input x. Hence the adversary has obtained key information.

A lot of block ciphers are inherently secure against this type of attack, such as most Feistel ciphers (another important family of block ciphers, see Sect. 2.1.3). In most cases, their key operation is performed prior to an S-box layer, and they do not employ a key whitening. However, most SPN ciphers, such as the AES, employ a key whitening to prevent a 'peel off' of the final permutation, others to increase the complexity of a *brute force attack* (key-search). These are vulnerable to this implementation attack. An example of the composite cipher DES-X [106], used in the Win2K encrypting file system, defined as:

$$\text{DES-X}_{k_0,k_1,k_2}(m) := DES_{k_0}(m \oplus k_1) \oplus k_2\,,$$

where key k_2 can be recovered from the implementation. A few possible strategies to prevent a key whitening attack are:

- Using block ciphers with *key dependent S-boxes*, such as Khufu [130] and Blowfish [168]. However, one should not directly modify S-boxes of other ciphers, since this leads to non-standard implementations that might contain unexpected weaknesses.
- Tweaking the cipher design, such that the S-boxes are different, but the input-output behavior of the cipher is equal. This can be achieved by masking techniques (inspired by techniques against side-channel attacks[7]), obfuscation techniques, or by generating S-boxes on the fly.
- Trivial leaks can also be prevented when suitable verification mechanisms are in place. These could be checksum computations to verify that the executable code has not been modified, checking that encryption and decryption behavior is matching, or performing trivial tests. However, current

[7] These are attacks that exceed the black-box model, as they take other information besides the functionality of the cryptographic primitive into account. See Sect. 2.3.2.

state-of-the-art obfuscation and tamper resistant software techniques are not able to protect these verification techniques. A seminal attack example in the area of tamper resistant software is the clone attack by Van Oorschot *et al.* [183].

- *White-Box Cryptography*, the topic of this thesis.

1.4 White-Box Cryptography

In this thesis, we address implementation issues of cryptographic ciphers in a white-box model. The research on cryptology in such a white-box model is called *white-box cryptography*, and implementations of cryptographic primitives for such a model are referred to as *white-box implementations*. Initially, we focus on white-box implementations of block ciphers, because of their widespread popularity and practical interest. Moreover, the initial work on white-box cryptography was conceived on block ciphers. Later, we extend our discussion towards other cryptographic primitives, such as asymmetric schemes and signature schemes.

The main research questions are: Is it *possible* to implement cryptographic primitives in a 'secure' way, even if the implementation is under full control of an adversary? And if so, which techniques can be used to construct such implementations? What would their impact be on the design of block ciphers, and on the deployment of cryptographic primitives in applications?

How would these constructions look like? And what would the implication of this be?

The two attack scenarios that are presented above, already indicate that novel implementation conventions will be required. Firstly, key information will need to be distributed over the whole implementation, to prevent trivial retrieval through static analysis. Secondly, randomness will need to be injected into the cipher operations, to prevent dynamic attacks and targeted tampering of the cryptographic implementation. Up to some extent, we still want to preserve the original input-output behavior of the block cipher. Therefore, white-box cryptography will mainly address strategies on how to compile a block cipher in a secure way.

1.5 Outline of this Thesis

This thesis is organized in six chapters. Following the introduction presented in this chapter, we give an overview of block ciphers and their design strategies. We

also discuss 'classical' cryptanalysis techniques and side-channel cryptanalysis techniques, as they are a source of inspiration for white-box cryptanalysis.

The research in white-box cryptography can be categorized into three main topics: practical constructions of white-box implementations; their formal definitions and the results based on theoretical foundations; and applications where this technology can be deployed. We will devote a chapter to each category.

- *Chapter 3: White-Box Implementations.* Firstly, we elaborate on the white-box model, and compare this to other attack models, such as the black-box model and the grey-box model. Secondly, we present and analyze the white-box implementations of the DES and the AES by Chow *et al.* [43, 42]. Our main contribution in this chapter is a cryptanalysis of white-box DES implementations. We also present a few conclusions and ideas for further research on this subject.

- *Chapter 4: A Theoretical Model for White-Box Cryptography.* In this chapter, we present a theoretical model for white-box cryptography, inspired by theoretical models of software obfuscation. In this new model, the possibility of secure white-box implementations is investigated. To achieve this, we define the concept of *security notions* and experiments, to capture how a cryptographic primitive is deployed, and what the strategy of the adversary is. Based on some *proofs by reduction*, we obtain a number of positive and negative possibility results.

The principle objective of these two chapters is to assess the security of white-box implementations. In Chapter 3, practical schemes are investigated and cryptanalysis techniques deployed, while in Chapter 4, a formal approach is described.

- *Chapter 5: Applications.* Cryptographic ciphers are a building block for secure applications. Although they often represent only a small component of the full application, compromise of a cryptographic piece can render the application insecure and often useless. In this chapter, we investigate what type of applications benefit from advance in white-box cryptography, and how white-box implementations can be securely integrated. More than often, they are a promising solution for problems that were believed to be unsolvable. This however also raises questions on the feasibility to design secure white-box implementations.

- *Chapter 6: Conclusions and Future Research.* Lastly, an overview is presented of the results that have been obtained in this thesis. This captures the achievements in the three topics described above, and elaborates on the interpretation of the results. Since white-box cryptography is to be considered still premature, we present an overview of directions for future research.

Chapter 2

Block Ciphers

2.1 Introduction

A block cipher is a pair of cryptographic algorithms, one for encryption, one for decryption. They are one of the more popular cryptographic primitives designed to enable concealed message exchange (Shannon [173]). In this chapter, we describe the design principles of block ciphers, present some of state-of-the-art block ciphers, and give an overview of cryptanalysis techniques.

2.1.1 Terminology

We will use the following terminology for a basic cryptographic system, as depicted in Fig. 1.1. The *sender* is denoted as the legitimate originator of a message, while the *receiver* is the legitimate recipient of a message. An *adversary* is an entity that aims at breaking into the communication between the legitimate parties. To conceal a message (*plaintext*) from an adversary, it is sent in garbled form (*ciphertext*) from the sender to the receiver. The ciphertext is the *encryption* of the plaintext, and is computed by the sender. The receiver is able to obtain the original message by *decryption* of the ciphertext.

Adversaries can be classified into two categories: *passive* adversaries only observe the communication between the legitimate parties in order to obtain information on the plaintext or key; *active* adversaries observe, tamper, and interact with the communication, in order to obtain information on the plaintext, to forge messages, or to recover key information.

2.1.2 Objectives and Definitions

The objectives of information security can be categorized into three main goals.

- *Confidentiality* – Concealing a message against unauthorized eavesdropping.
- *Integrity* – Protecting a message against tampering.
- *Authentication* – Refers to *entity authentication*, related to the identification of the (legitimate) parties, whereas *data authentication* is equivalent to integrity.

Depending on the application in which a cryptosystem is deployed, a number of other objectives can be formulated, such as *non-repudiation*,[1] and *availability*.

A cryptosystem is designed in order to achieve these objectives, and a cryptographic *cipher* is a pair of algorithms that implements the encryption (E) and decryption (D) primitives for a cryptosystem. In accordance to Kerckhoffs' principle [104], the algorithms are public, while a (secret) key is used to instantiate a cipher. The main classification of ciphers was established by Diffie and Hellman [59], and relates to the concept of *key*.

- A *symmetric cipher* is a function family (E, D), associated by a keyspace $\mathcal{K}$.
$$E : \mathcal{K} \times \mathcal{P} \to \mathcal{C} \qquad D : \mathcal{K} \times \mathcal{C} \to \mathcal{P}\,.$$
$\mathcal{P}$ is the plaintext space and $\mathcal{C}$ the ciphertext space. (E_k, D_k) denotes an instantiation of the cipher with the key $k \in \mathcal{K}$. The encryption function E_k is injective[2] and may be randomized. The decryption is defined such that it satisfies the correctness condition: $\forall m \in \mathcal{P}, D_k(E_k(m)) = m$.
- An *asymmetric cipher* is a triple (E, D, G); (E_K, D_k) a particular instantiation. G is the key generation function, that produces a public encryption key K, and the corresponding private decryption key k for some seed s (chosen at random by the receiver).
$$G(s) \to (K, k) \qquad E_K : \mathcal{P} \to \mathcal{C} \qquad D_k : \mathcal{C} \to \mathcal{P}$$
The algorithms are designed such that $\forall m \in \mathcal{P} : D_k(E_K(m)) = m$.

[1] *Non-repudiation* refers to a guarantee that an entity cannot deny to have sent or received a message.

[2] A function $F : X \to Y$ is injective, if $\forall x_1, x_2 \in X$: if $F(x_1) = F(x_2)$, then $x_1 = x_2$. This is often referred to as *(left) invertible*.

Symmetric ciphers are typically more efficient than asymmetric ciphers, but can only be used in situations where the secret key can be a priori established between the parties. In the case of asymmetric ciphers, all parties (legitimate and non-legitimate) are able to encrypt using the public key K, while decryption is only possible for the receiver (with the knowledge of the private key k). The assumption is that an adversary should not be able to compute k, given K (and G). The dual scenario of public key encryption is *digital signatures* where only one party is able to sign (the dual of decryption), while all parties are able to verify a signature (the dual of encryption). We refer to Menezes *et al.* [129] for more details on asymmetric cryptography and digital signatures.

Symmetric ciphers can be divided into *block ciphers* and *stream ciphers*.

Definition 1 *A* **block cipher** *is a keyed family of pseudo-random permutations, that operates on a fixed length input and produces a fixed length output. It is stateless, time-invariant, and instantiated by a secret key k. For each key, we obtain a permutation that is independent of all other instances. We define a block cipher with* block size *n as a block cipher with binary string input-output words of size n.*

$$BC\ encryption_k : \{0,1\}^n \rightarrow \{0,1\}^n$$

To allow encryption on plaintexts of arbitrary lengths, *padding schemes* and *modes of operation* are introduced. Padding schemes introduce padding bits that are added to plaintext messages with a smaller size than the block size. The encryption operation becomes $m \rightarrow E_k(m||\text{padding})$. Plaintext messages of larger size are divided into n-bit words (the last word possibly being padded). The block cipher will operate on each of these words, according to a specified *mode of operation*. The resulting ciphertext is the concatenation of the output words. As an example, the *cipher-block chaining (CBC)* mode is depicted in Fig. 2.1.

To make each message unique, a secret random *initialization vector* (IV) is introduced at the beginning of this mode of operation, hence randomized encryption is obtained. Formally, CBC can be described with the following equation: $c_i = E_k(m_i \oplus c_{i-1})$, where $c_0 = \text{IV}$. Many other modes of operation have been introduced, including

- *electronic code book (ECB)*: $c_i = E_k(m_i)$,
- *cipher feedback (CFB)*: $c_i = E_k(c_{i-1}) \oplus m_i; c_0 = \text{IV}$,
- *counter mode (CTR)*: $c_i = E_k(IV \| i) \oplus m_i$.

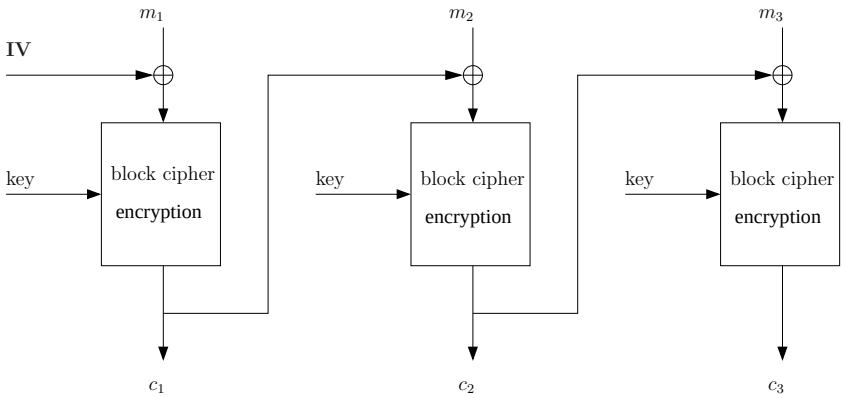

Figure 2.1. Cipher-block chaining (CBC) mode of operation

2.1.3 Design

According to Definition 1, a block cipher is idealized by a pseudo random permutation. This means that is it should indistinguishable from a random permutation on n bits. I.e., a permutation that is chosen at random according to a uniform distribution from the family of all permutations on n bits. To achieve *perfect* security, block cipher implementations would need to be able to implement any given permutation. But this is infeasible for reasonable block sizes, since there are $2^n!$ permutations on n bits. Instead, block ciphers implement only a fraction of the all possible permutations, instantiated by a secret key. For an m-bit key size, a family of block ciphers implements 2^m different permutations. Consequently, block ciphers can achieve at most *computational security*. A block cipher is called computational secure, if an adversary cannot distinguish an instantiation with computationally less effort than a brute force search over the cipher's key space.

Unfortunately, (most) block ciphers do not come with a proof of security. Rather, the security of block ciphers is based on *ad hoc security*, which relates to the unsuccessful cryptanalysis of the cipher by many cryptanalysts for many years. As techniques in cryptanalysis improve, they impose more requirements on the design of block ciphers. For example, the discovery of differential cryptanalysis, initiated the study of the design of S-boxes and diffusion layers with important differential cryptanalysis properties [1].

In [174], Shannon described the concept of *product ciphers*. These ciphers consist of several key-dependant rounds, that combine simple operations. By itself, one single round does not provide any security, but together, they provide a complex network of operations that is 'easy' to analyze, but (hopefully) difficult

to cryptanalyze. A complex design is avoided, because these might be slow in software, or require a lot of gates in hardware.

Two methods were suggested by Shannon in [174, page 708] to frustrate cryptanalysis, *Diffusion* and *Confusion*, which we present in our own words:

Diffusion. The diffusion property is a quantitative notions that refers to the dependence on each plaintext and key bit to each ciphertext bit. When a cipher has a good diffusion, any difference should propagate well. In particular, flipping one bit of the plaintext should cause each output bit to flip with a probability of one half. The propagation property is often captured by the *avalanche effect*. Typical operations that are deployed for block ciphers are permutations and wide linear operations. These refer to functions with a large input and output domain. For efficiency reasons, diffusion operations are preferred to be simple.

Confusion. This is qualitative notion that refers to making the relationship between the plaintext bits, the ciphertext bits, and key bits as complex as possible. Non-linear operations achieve confusion. Usually, they consist in the parallel application of high degree functions called substitution boxes, or S-boxes. In software, these S-boxes are usually implemented as lookup tables. Since storing an m-bit to n-bit S-box requires $n \cdot 2^m$ bits of storage, S-boxes are kept small, typically with an input size (m) of 4 to 8 bit.

Product ciphers use alternating substitution and diffusion operations to achieve both confusion and diffusion respectively. Two families of iterated block cipher networks are commonly found in modern cryptographic ciphers: *Feistel* ciphers, and *Substitution-Permutation Network* (SPN) ciphers.

Feistel ciphers have the following properties:

- Typically, they apply a complex operation on half the input bits. Denote with X_i the input to round i; L_i, and R_i respectively the left and right half of the input ($X_i = L_i||R_i$). Then, the output of the round can be described as $X_{i+1} = R_i||(F(k_i, R_i) \oplus L_i)$, where k_i is the round key, derived from the original key k via a key scheduling algorithm, and F some fixed function that involves the round key. Fig. 2.2 describes one such a round.

- Given the output of a round, and the round key, the original input can be computed as follows: $X_i = R_{i+1} \oplus F(k_i, L_{i+1})||L_{i+1}$. Hence, the F function need not to be bijective. It is advisable though, that the function

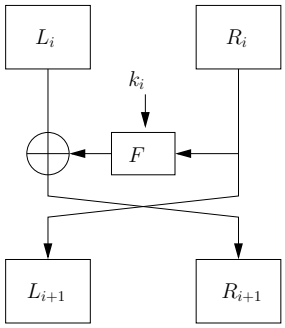

Figure 2.2. One Feistel cipher round

F is "close" to surjective. Otherwise, attacks as described in [156] become possible.

- Due to the symmetry between encryption and decryption, both functions can be implemented with the same algorithm; only the round keys needs to be computed in reverse order. Hence, as hardware implementations, their circuits can be re-used, resulting in a smaller footprint. This is one of the main reasons why Feistel ciphers are still a popular design strategy. On the downside however, Feistel ciphers need a substantial number of rounds to comply with the diffusion requirement, since a round function transforms only half of the bits.

Modern Feistel ciphers might slightly deviate from this design, but hold similar properties. The most common example of a Feistel cipher is the Data Encryption Standard (DES), which is the first widely accepted modern cipher. We introduce DES in Sect. 2.2.1, and discuss implementation issues and its white-box implementations in Chapter 3 of this thesis.

Substitution-permutation network (SPN). These ciphers transform all the input bits of the round. They have the following properties:

- Typically, the round function of a SPN cipher consists of three layers: key addition, a substitution (confusion) layer (several S-boxes in parallel), and a permutation (diffusion) layer.
- Every building block of the cipher needs to be invertible, in order for the round functions to be invertible, to allow decryption.
- Encryption and decryption require different algorithms, because the layers need to be processed in reverse order.

The Advanced Encryption Standard (AES) is a SPN cipher, which we introduce in Sect. 2.2.2.

2.2 Block Ciphers

2.2.1 The Data Encryption Standard (DES)

The *DES* [142] is a Feistel block cipher that operates on 64-bit blocks and uses a 56-bit key; it has been adopted by NBS (currently named NIST [139]) in 1977 as the encryption standard with the perspective of being fast, simple and secure, and has enjoyed widespread use ever since. It is the first widely accepted modern encryption algorithm with open and fully specified implementation details. Therefore the DES has been subjected to intense academic scrutiny, and hence motivated the modern understanding of block ciphers and their cryptanalysis.

The DES has been the most widely used cryptosystem in the world, but is now considered to be insecure for many applications. This is due mainly to its small key size (of 56 bits) and several attacks that have been presented [16, 124, 149]. In 1998, the Electronic Frontier Foundation created the $ 250,000 "Deep Crack" [62] machine to crack DES-encrypted messages in about 56 hours of brute-forced search for the secret key. In 2006, the COPACOBANA [178] machine was built, with 120 low-cost FPGAs. It is commercially available, costs about $ 10,000, and is able to perform an exhaustive key search on the DES in under 6.4 days on average.

Although its successor, the *Advanced Encryption Standard (AES)*, is designed to meet new security requirements and the evolution of computers, the DES or variants such as 3DES[3] are still implemented and deployed in various applications, in particular in the financial sector for banking transactions and ATMs. ATMs are difficult to replace frequently, and have a lifespan of about eight years. The DES has been replaced on ATMs between 1998 and 2006 by 3DES, and updating to another block cipher such as the AES is a process that will take about a decade to implement [4].

Design of the DES

The DES consists of 16 Feistel rounds, accepts as input a 64-bit input plaintext, and produces the corresponding ciphertext under a 56-bit key. The input to each round r is divided into a left (L_{r-1}) and a right half (R_{r-1}) of each 32 bit. The right half bits serve as input to the function F; its output is XOR-ed with the left

[3] 3DES or *Triple-DES* is a variant of the DES, designed to enlarge the key space, without a need to switch to another cipher implementation. The simplest variant, with a 192-bit key, is described as $\mathrm{DES}_{k_3} \circ \mathrm{DES}_{k_2}^{-1} \circ \mathrm{DES}_{k_1}$. For the 112-bit key variant, $k_1 = k_3$.

half. At the end of the round, the two halves are swapped. Figure 2.3 gives an overview of the DES, and depicts one such round in more detail. The 16 rounds are insulated by an initial permutation IP at the beginning, and its inverse IP^{-1} at the end.

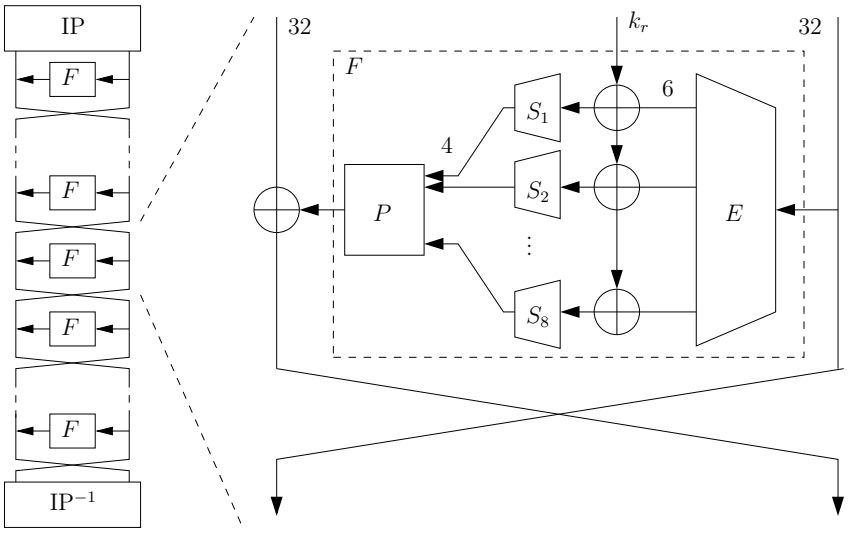

Figure 2.3. the Data Encryption Standard

The F-function is the non-linear function that exhibits the confusion and diffusion properties which good cryptographic ciphers require, and depends on a 48-bit round key that is computed from the 56-bit key using the DES key schedule algorithm. Firstly, the 32 input bits are expanded to 48 bits through the expansion operation E, which replicates 16 bits. The outcome is bitwise added with the 48-bit round key k_r, and subsequently evaluated by the 8 S-boxes $S_1, S_2, \ldots, S_8$. Each S-box is a non-linear mapping of 6 onto 4 bits. Finally, the resulting 32 bits are permuted through the Permutation block P.

A formal description of the DES is presented by Algorithm 1.

The DES has been designed with the perspective of being simple and secure, and fast and compact in hardware (which was expensive in the seventies). The DES key scheduling algorithm is designed to be functionally equivalent to the identity function after 16 rounds. Hence its output can be fed into the key schedule function again for the next encryption, such that there is no need to store the key in a seperate register.

Algorithm 1: Formal description of DES

input : plaintext m
$(L_0||R_0) := \mathrm{IP}(m)$
for $r : 1 \dots 16$ **do**
  $L_r := R_{r-1}$
  $R_r := L_{r-1} \oplus P((S_1||S_2|| \dots S_8)(k_r \oplus E(R_{r-1})))$
end
output: ciphertext $c := \mathrm{IP}^{-1}(L_{16}||R_{16})$

Relevance of the study on the DES

Although the DES is in withdrawal phase, it is still an important study object. First of all, from a cryptanalytic point of view, it is interesting to mount attacks on the DES in order to compare with other attacks. Linear and differential attacks have extensively been deployed on the DES. Moreover, the design principles of the DES are used in many other ciphers, including Triple DES, TEA [187], Twofish [169].

From a theoretical perspective, Feistel ciphers are an interesting study subject. In [120], Luby and Rackoff presented a construction of a block cipher that is information theoretically secure [72], given a random function. In practice, given a good round function, four rounds of Feister are sufficient to achieve indistinguishability from a random permutation under *chosen plaintext attack* (CPA).

The DES has been the first cipher for which a white-box implementation is presented. We elaborate on white-box DES implementations and subsequent cryptanalysis in Chapter 3.

2.2.2 The Advanced Encryption Standard (AES)

Due to the small key size and the slow software performance of the DES, NIST announced a competition for a new standard, the *AES* [138]. In 2001, after a five-year competition and standardization process, the Rijndael block cipher [54], fixed to a block size of 128 bits, was selected. Nowadays, it is one of the most popular algorithms used in symmetric cryptography.

Unlike the DES, AES is a substitution-permutation network, not a Feistel network. It is fast in both software and hardware,[4] easy to implement, and requires only little memory.

[4]Recently, Intel announced that it will include instruction sets in their 2009 processors to speed up AES computations [98].

Description of the AES

The Advanced Encryption Standard (AES) is a substitution-permutation network that supports key sizes of 128, 192, and 256 bits (denoted as AES-128, AES-192, and AES-256 respectively). It works on a fixed block size of 128 bits (whereas Rijndael, the original submission, can be used with key and block sizes in any multiple of 32 bits, with a minimum of 128 bits and a maximum of 256 bits).

Depending on the key size (128, 192, or 256), AES consists of respectively 10, 12, or 14 rounds, that operate on a 4×4 array of bytes (called state). A round is specified in four steps as follows:

1. `SubBytes` – evaluate each byte of the state by the same S-box, 16 times in parallel; The S-box is defined by the equation $S(x) = M(1/x) + b$ over the field $\mathrm{GF}(2^8)$ where M is suitably chosen and b is a constant. Such a definition gives tight linear and differential bounds.
2. `ShiftRows` – a cyclic shift of each row, which leaves the top row of four bytes unchanged.
3. `MixColumns` – mixing of the four bytes of a column, by multiplication of each column with a constant matrix over the field $\mathrm{GF}(2^8)$.
4. `AddRoundKey` – binary addition of the state with the 128-bit subkey;

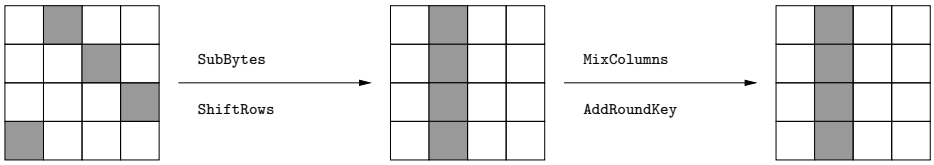

Figure 2.4. A schematic overview of one round of the Advanced Encryption Standard

A preliminary `AddRoundKey` step is performed before the first round (with a whitening key), and `MixColumns` is omitted in the final round. Note that the MixColumns of the last round has no importance, as when it is applied, an attacker can cancel this operation by applying a known transformation.

The round operation is designed as such that a change on the input to a round can potentially affect all of the output bits after only two rounds. The key schedule of AES-n takes an n-bit key and transforms it into NR+1 subkeys

of 128 bits each, where NR denotes the number of rounds. We refer to [54] for further details on AES.

2.3 Cryptanalysis

The ultimate goal of the cryptanalyst is to present an attack faster than any generic attack on the block cipher, i.e., faster than exhaustive key search. In the exhaustive key search attack, the attacker tries all the keys sequentially, and tests whether the intercepted messages are being decrypted into something with redundancy. The assessment of the security of block ciphers is the main motivation of research in the field of block cipher cryptanalysis. This process of security justification consists of applying cryptanalytic techniques to the cipher, where each technique aims at exploiting different design flaws. Through this process, the cryptology community learns to understand the design methodologies that could yield 'secure' block ciphers; i.e., block ciphers that resist all known attacks.

In this thesis, our objective is to assess the security of block cipher implementations in a *white-box attack context*, which exceeds any other attack model studied before. Nevertheless, cryptanalysis techniques that have been developed in the past, often apply to white-box implementations or are a source of inspiration for white-box cryptanalysis techniques. In this thesis, we present an overview of the most important cryptanalytic techniques.

2.3.1 Black-Box Cryptanalysis

Black-box cryptanalysis refers to cryptanalysis techniques in the *black-box attack model*. This is the traditional attack model that is studied in cryptography, and is depicted in Fig. 2.5. In this model, the (black-box) security of a cryptographic primitive is captured by using a *security notion* where an adversary is given black-box (oracle) access to the functionality. That is, the adversary can send a number of queries to the oracle, each representing a plaintext or a ciphertext. The oracle answers with the corresponding ciphertext, encrypted by the cipher E under the (secret) key k. The adversary is not able to obtain any other information besides the functionality of E_k. This captures the concept of an *active adversary* in the basic cryptosystem.

The objective of the adversary is to gain *some knowledge*. This can be quite broad, e.g., obtain information of (part of) the key, find the original plaintext message for a specific ciphertext, or distinguish the oracle (that implements the

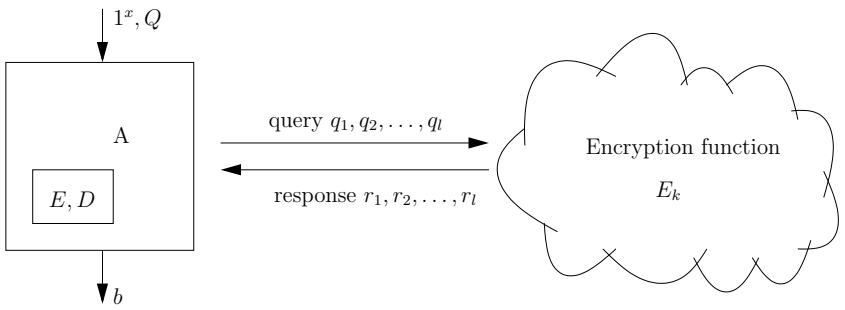

Figure 2.5. The black-box attack model

cipher) from a random oracle (that implements a random function). The question that is posed to the adversary is captured by Q, its answer by b. Moreover, A has the functions E and D as subroutine, since these are public in accordance to the Kerckhoffs principle, while k is unknown. 1^x indicates the size of the input. We elaborate on these formal models and security notions in chapter 4.

A cryptographic primitive is said to be secure under a certain model, if it resists an adversary with appropriate powers. We can distinguish between several types of attacks.

Generic Attacks are attacks that are applicable to any cipher without any pre-condition, hence independent of their design. The most common generic attack is exhaustive key search. This attack can work under a ciphertext-only assumption, if the redundancy of the plaintext is sufficiently high, or a known plaintext assumption otherwise. Another generic attack is the tabulation attack, where an attacker obtains a table containing the encryption of a given plaintext under all keys. Once the ciphertext of this plaintext is observed by the attacker, he can access the pre-computed table and find the key in one table lookup. Hellman suggested that there are many time-memory tradeoffs possible applying to any block cipher [87]. The work of Hellman was further improved in a series of works [29, 141], suggesting other tradeoffs or applications of these results.

Statistical attacks aim at finding statistical properties that distinguish between a given cipher (or a reduced-round version of it), and a random permutation or function. These include differentials, or linear approximations (finding correlation between plaintexts and the ciphertexts). There are several statistical properties that are used in the analysis of block ciphers. An example is the relation between the parity of a subset of plaintext bits and a subset of ciphertext bits (which is investigated in linear cryptanalysis [124]). Some statistical properties combine

information between more than one plaintext/ciphertext pair, and are used in e.g., differential cryptanalysis [17].

Differential cryptanalysis was the first theoretical attack that could break DES faster than exhaustive key search. It was introduced in 1990 by Biham and Shamir [16].

The attack exploits the relation between the differences of plaintext pairs $(P, P+\Delta P)$ and the differences of the corresponding ciphertext pairs $(C, C+\Delta C)$, where ΔX denotes the *difference* between the string X and X', and is defined as $\Delta X = X + X'$ for some operation $+$ over these strings. When the $\oplus$ (exclusive addition, or xor) operation is used to combine round keys with the internal data, a difference defined over $\oplus$ is unharmed by the round key addition.

The attack tries to find *differentials* $\Delta P \rightarrow \Delta C$ that occur with a significant larger probability than 2^{-n}, where the probability of a differential is defined as

$$\Pr(\Delta C|\Delta P) = \Pr_{P,k}\left[E_k(P + \Delta P) = E_k(P) + \Delta C\right],$$

where $\Pr_{P,k}$ is an average probability over P and k.

Later, the attack was generalized to deal with partial differences (called truncated differentials) and sets of plaintexts (instead of only pairs) in higher-order differentials [108] . The methodology of differential cryptanalysis is employed in our cryptanalysis of the white-box DES implementation in Sect. 3.8.

Linear cryptanalysis was discovered by Mitsuru Matsui in 1993 [124], although the fundaments of its principle were initiated earlier with the analysis of FEAL [177, 126]. In general, linear cryptanalysis is based on finding *linear approximations* for a cipher. The attack relies on a statistical bias of the parity of the bits of the plaintext, ciphertext and key of the cipher. The bias ϵ over a parity λ_P, λ_C is defined as

$$\epsilon = \Pr\left[\lambda_P \cdot P = \lambda_C \cdot C\right] - \frac{1}{2},$$

where $\cdot$ denotes the scalar product. Using this technique, Matsui has experimentally broken DES in 1994 [125]. Specifically, Matsui exploited the linear correlation in the fifth S-box that was discovered by Shamir [171]. Although S-boxes are balanced by design, the discovery of biased S-boxes enables a point to mount a linear attack. If one can find a linear relation for a block cipher that holds with a probability of $\frac{1}{2} + \frac{1}{N}$, then one can expect to be able to recover keybits with about N^2 plaintexts. If the best linear approximation implies that N^2 is greater that the number of possible plaintexts 2^n, the block cipher is considered secure against linear cryptanalysis.

Algebraic attacks on block ciphers are a recent technique, introduced by Courtois *et al.* [51]: they treat a cipher as an overdefined set of low degree equations, which an attacker tries to solve. Each cipher can be described as a set of multivariate equations in the input bits, output bits, and key bits. If these equations are of low enough degree, or have some specific structure, an attacker can find the key bits from a small sample of plaintexts and ciphertexts by solving the set of equations.

Other classes of attacks include *meet-in-the-middle* attacks and its variants. They are one of the first cryptanalytic techniques that were published. E.g., on Double-DES with two independent keys, there is a meet-in-the-middle attack [184]. *Related key attacks* [13] refer to attacks where an adversary analyzes a cipher under different keys, where a relationship between the keys is known (although they keys themself are unknown). To be feasible to deploy however, an adversary must be able to observe the functionality of a cipher under several different keys.

Reduced round attacks introduce an iterated process of cryptanalysis, where an attack is first deployed on a reduced number of rounds of a cipher, which might provide insight to extend the cryptanalysis to a larger number of rounds or even the full round block cipher. E.g., for AES-128 and AES-192, a 7 and 8 round cryptanalysis has been presented respectively (out of 10 and 12 rounds) [107]. For the DES, three theoretical attacks already exist on the full 16 rounds, but until now, no practical attacks have been found on the full round version.

2.3.2 Side-Channel Cryptanalysis

In the past decade, the field of cryptanalysis has experienced major changes. Not only do cryptanalysts now investigate the design of cryptographic primitives, but they also examine their implementation into devices. In 1995, Paul Kocher discovered *implementation attacks*, and showed how timing information of cryptographic operations can be used to break the implementation of several cryptosystems [109]. This class of attacks is referred to as *side-channel attacks*. The term *side channel* is an abstraction of all unintended information leakage due to implementation as a circuit.

Side-channel analysis is a powerful cryptanalytic technique that can be used to exploit data-dependent physical leakage in order to recover secret data from actual implementations. In Fig. 2.6, a side-channel attack model is depicted.

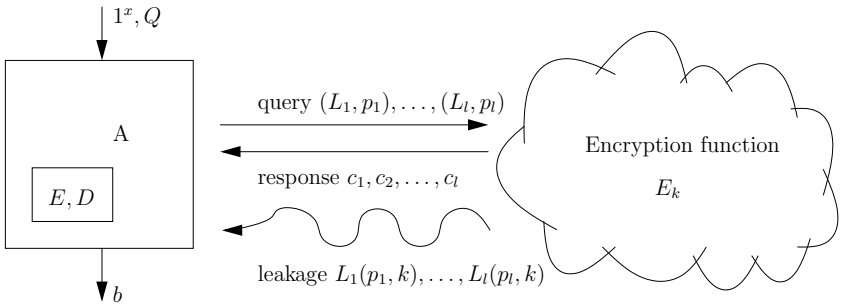

Figure 2.6. The grey-box attack model

The main difference between the side-channel attack model and the black-box attack model, is the presence of a leakage function L. This leakage function abstracts information an adversary is able to obtain due to the implementation of the algorithm as a circuit, and can originate from diverse physical characteristics. Not only does this leakage depends on the intended input (p_i), it is also influenced by other factors such as physical characteristics of the device in which the circuit is implemented, and external environmental conditions such as heat, radiation, etc. These external conditions can be manipulated by the adversary and are captured in the model within the queries (L_i, p_i).

Information leakage functions

Side-channel attacks were introduced with the discovery of *timing attacks* by Paul Kocher [109]. These attacks rely on the fact that some operations take more time to compute than others. Hence, when a correlation exists between the secret key, and the number of such operations, often a timing attack can be deployed. Countermeasures against these attacks are fairly easy to implement when introducing delay operations, but typically they reduce the efficiency of an implementation.

Simple and Differential Power Analysis (SPA and DPA respectively), published in 1998 by Kocher *et al.* [110], are a powerful class of non-invasive side-channel attacks. These attacks use measurements of the device's power consumption to disclose secret key material. *Simple Power Analysis* involves the interpretation of power consumption by a circuit in time. A typical attack on cryptographic circuits would be on RSA implementations, where the key bits decide whether a multiplication or addition operation will be performed. Due to its difference in power consumption, the correlation between the key bits and power consumption allows to mount a simple power analysis attack [140]. *Differential*

Power Analysis is a more advance technique that involves statistical analysis of power consumption information obtained by multiple measurements.

Electromagnetic emanation was introduced as a side channel in 2001. Fundamental works in this area were published by Quisquater and Samyde, as well as Gandolfi, Mourtel and Olivier [151, 67]. *Template Attacks* were introduced by Chari *et al.* [41] in 2002.

Side-channel attacks can be categorized into two classes: *passive attacks*, which are non-invasive, such as power consumption and timing attacks; and *active attacks*, which involve tampering of the circuit, such as *fault attacks*. The complexity of side-channel attacks is expressed not only in computational complexity (as with black-box attacks), but also the quality of the side-channel information (related to the presence of noise).

Theoretic approaches

Side-channel attacks are categorized according to the type of leakage function, the quantity of the side-channel information, and their quality. The leakage functions available to an adversary and their conditions are described in a model. A cryptographic primitive is said to be secure under a certain model, if it resists any adversary with appropriate powers. Obviously, it is desired that such a model is a reflection of the reality in which the cryptographic primitive is to be deployed.

Several side-channel models have been described in the past years [112, 167]. An interesting new research field within the context of theoretical modeling is *Physically Observable Cryptography*, introduced by Micali and Reyzin [132] in 2004. The approach is three-fold: (1) specify the physical world with a set of axioms (model), (2) define what is understood under the security of an algorithm, and (3) prove that no adversary for a secure algorithm can exist under reasonable assumptions. A substantial result within this context, is the design of *Private Circuits I* by Ishai *et al.* [100], who prove that secrecy can be guaranteed when an adversary has access to a bounded number t of wires in a circuit. *Private Circuits II* [99] persues a similar goal in the presence of tamperable circuits. However, the main issue is how realistic these models are. The results are rather theoretic, because the resulting (secure) implementation is rather big: $\mathcal{O}(nt^2)$ gates, where n denotes the size of the original (insecure) circuit. Also, once an adversary is able to probe t wires, the effort is small to probe $t+1$ wires, in which case the implementation becomes insecure. Moreover, as the size of the circuit increases, more information becomes potentially observable for other side-channels such as power analysis.

2.4 Conclusion

We presented block ciphers, which are widespread cryptographic primitives that are of particular interest for white-box cryptography. Traditionally, block ciphers are designed to withstand black-box attacks, and their security is assessed by a process of scrutinizing. This include the deployment of cryptanalytic techniques such as differential cryptanalysis and algebraic cryptanalysis. A similar approach will be conceived to assess the security of white-box implementations.

In practice however, popular trusted ciphers like RSA and AES might not provide the level of security as expected, since they are not designed to operate in environments where their execution could be observed. The introduction of side-channel cryptanalysis techniques showed that even conditional access might suffice for an adversary to obtain confidential information such as secret keys. The natural process for research in side-channel cryptology exists in defining a attack model (e.g., an adversary is allowed to probe t wires), and develop countermeasures within that model. However, it turns out to be difficult to define a model that captures real-world adversaries. Hence, the following natural problem arises:

Problem 1 *Given a cryptographic primitive that is secure against adversaries in a given model. What if an adversary does not comply any more to the model? What if an adversary 'steps out' of the model?*

This is one of the issues that White-Box Cryptography aims to address.

Chapter 3

White-Box Implementations

3.1 Introduction

The main objective of this dissertation is to assess the security of cryptographic primitives in the presence of a white-box adversary. This research is denoted as *white-box cryptography*. In this chapter, we analyze the security of practical implementations of such primitives, denoted as *white-box implementations*. We refer to the methods and procedures of such an adversary as *white-box cryptanalysis*.

The main interest of white-box cryptography lies in the implementation of symmetric ciphers, in particular block ciphers. In this chapter, we assess the security of these techniques, by analyzing the implementations and deploying cryptanalysis techniques. This is similar to the 'proof by scrutiny' approach (see Sect. 1.1.4). In Chapter 4 we will extend the research towards arbitrary cryptographic primitives, and assess white-box security based on 'proof by reduction'. After a discussion on the white-box attack context, and introducing the main obfuscation strategy, we present an overview of the state of the art in white-box cryptography. One of our main contributions is the cryptanalysis of encoded white-box DES implementations. We also initiate an analysis of basic building blocks of block ciphers, and conclude with strategies towards secure white-box implementations by formulating 'white-box design criteria' for block ciphers.

3.1.1 Attack Models

One can distinguish three main attack models that capture the attack capabilities of an adversary on cryptosystems. These are the following:

- The *black-box model* is the traditional attack model, where an adversary has only access to the functionality of a cryptosystem.

- A *grey-box model* refers to a model where a leakage function is present. In such an attack context, the adversary can deploy side-channel cryptanalysis techniques. Due to the large variety of leakage functions (see Sect. 2.3), several grey-box models can be defined.

- In the *white-box model*, the adversary has total visibility of the software implementation of the cryptosystem, and full control over its execution platform. One could refer to the *white-box model* as the worst-case model, where in contrast to grey-box models, it is impossible for an adversary not to comply with the this model. The white-box model is used to analyze algorithms that are running in an untrusted environment, that is, an environment in which applications are subject to attacks from the execution platform.

In 2002, Chow *et al.* [42] introduced the *White-Box Attack Context (WBAC)*, which assumes that:

- fully-privileged attack software shares a host with the cryptographic software, having complete access to the implementation of algorithms;

- dynamic execution (with instantiated cryptographic keys) can be observed;

- internal details of cryptographic algorithms are both completely visible and alterable at will.

The main goal of the adversary will be to extract secret key information. This definition also captures the *malicious host attack context* studied by Sander and Tschudin [162, 163] and Hohl [94].

The attack tools that are at the disposal of an adversary are diverse. They include analysis of the memory and implementation binary (e.g., to deploy an entropy attack; see Sect. 1.3.1.); interception of calls to the CPU or external libraries; use of debugging and analysis tools, such as IDA Pro [90] or Syser [176]; reverse-engineering; tampering of the software implementation, and fault injection at execution time. We refer to Main and van Oorschot [123] for an overview of software attack tools.

3.1.2 Comparison between Attack Models

In Fig. 3.1, an overview of the attack models is depicted. In a white-box model, black-box attacks can be deployed as well, since the functionality (by executing the given implementation) of the cryptosystem is available to the adversary. Hence, to be potentially *white-box secure*, a cryptosystem needs to be black-box secure in the first place.

Furthermore, when a cryptosystem is secure in the white-box model, it will be secure in any grey-box model as well, because the behavior of leakage functions depends on the operations of the implementation, which are completely visible in a white-box model. Hence, any leakage function can be simulated by a white-box adversary. As a result, secure white-box implementations are not only secure against the current state of the art side-channel attacks, but also inherently secure against *any future* side-channel attacks.

White-box attacks supersede side-channel attacks. However, there is still a significant gap between the two models. In grey-box models, adversaries are limited in observations to physical boundaries: accuracy of measurements, speed of logic-operations, noise, etc. These are boundaries that are not present in a white-box model. Moreover, software implementations are not bound to any platform, while circuit implementations are rather difficult to clone.

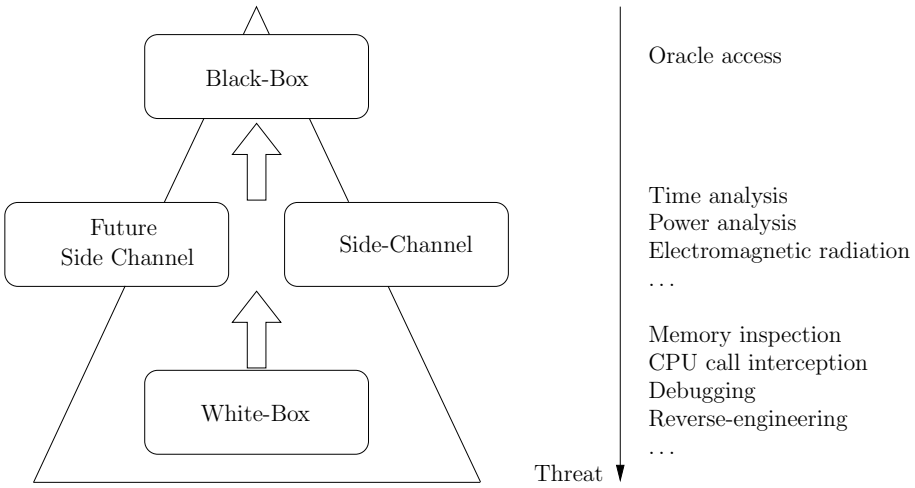

Figure 3.1. Attack model comparison

However, given this relationship, one can not make any direct connection between the security of an algorithm (in the black-box way), and the success probability of a side-channel or white-box attack. The latter two attack classes are only specific to the implementation of the algorithm.[1]

[1]Remark though, that in Sect. 3.9, we will make some indirect statements on this relation, based on cryptanalysis results on the state of the art of white-box implementations. Namely,

Numerous examples from side-channel analysis results show that, when nothing stands in between an adversary and the internals of the implementation, the choice of implementation is the sole remaining line of defense [19, 23, 40, 53, 52]. Therefore, 'white-boxing' refers to the methodologies used to *compile* a cryptographic algorithm into a secure implementation. Hence, we denote this as an obfuscation technique (but then, a very specific one).

3.1.3 Objectives of WBC

White-Box Cryptography, when it was proposed in 2001 by Chow *et al.* [43], was originally defined as an obfuscation technique with the following objective.

Definition 2 (Chow *et al.* [42]) White-Box Cryptography *is an obfuscation technique intended to implement cryptographic primitives in such a way, that even an adversary who has full access to the implementation and its execution platform, is unable to extract key information.*

The protected implementation should be a functionally equivalent program in which the key is no longer visible.[2] The techniques introduced by Chow *et al.* [43, 42] are implemented in the context of encryption algorithms. Ideally, a white-box encryption implementation should be so robust against analysis that an adversary would be inclined to resort to a plaintext/ciphertext pairs attack, as if the white-box implementation were a black-box.

However, there is a fundamental problem with Definition 2. Adversaries typically try to tackle the weakest link in an application. In some applications, isolating the cryptographic code in the application implementation might be sufficient to defeat its purpose. This is referred to as *code lifting*, and is a common threat to DRM applications. In some sense, white-box cryptography could also be seen as a technique to *hide a key in an even bigger key.* As such, protection against key recovery becomes a meaningless notion.

Therefore, we propose a new definition of the objectives for white-box cryptography, to make more sense in practice. We formalize and discuss this definition in detail in Chapter 4.

Definition 3 *The objective of* White-Box Cryptography *is to implement cryptographic primitives in such a way that, within the context of the intended application, having full access to the cryptographic implementation does not present*

that for some families of block ciphers, the particular building blocks that are designed to withstand black-attacks, are exactly those building blocks that impose a vulnerability to attacks in the white-box context.

[2] For practical purposes however, often an encoded version will be obfuscated. The result is a program that is functionally equivalent to the encoded version of the original program.

any advantage for a computationally bounded adversary in comparison to the adversary dealing with the implementation as a black box.

We admit that this definition is rather vague. Yet it allows a more precise definition of the requirements for a specific white-box implementation. What we suggest, is that the implementation of cryptographic primitives should be considered within the context of an associated *security notion*. A *security notion* is used in cryptography to capture the security requirements of a cryptographic primitive. It describes the capabilities of the adversary, and constitutes what a succesfull attack means (see also Sect. 4.3). A similar issue was pointed out by Hohenberger *et al.* [93], who presented a definition coined *secure obfuscation*.

Definition 2 refers to the *key recovery (KR)* security notion, which is an essential standard objective in cryptography. Another security notion would be for example *PR-CPA*, where an adversary tries to compute the plaintext corresponding to a given ciphertext, given access to the encryption functionality.

Note that key recovery implies invertibility of the cryptographic primitive (because due to the Kerckhoffs' principle, the corresponding decryption routine is available). The inverse is not always true.

$$\text{PR-CPA secure} \Rightarrow \text{KR-CPA secure.}$$

Hence, the non-invertibility security notion is a stronger requirement, but also harder to meet. For practical deployment of white-box implementations however, this security notion is much more interesting, sometimes even essential, for example when white-box implementations are deployed as an asymmetric primitive. We discuss the issue of running white-box implementations backwards in Sect. 3.7. Other security notions that might be relevant within the context of white-box cryptography include: prevent key recovery; non-invertibility; (targetted) tamper prevention; and traceability. We discuss these notions and their practical implications in Chapter 5.

Definition 3 opens a broad perspective on the concept of 'security' for white-box implementations. We formally describe this into a formal model for white-box cryptography in Chapter 4.

3.2 Obfuscation Strategy

When the internal details of an algorithm are exposed to an adversary, the last line of defense is the way in which the algorithm is implemented. This conclusion has been formulated in the study for secure implementations of cryptographic primitives on smart cards by Daemen *et al.* [52, 53], Biham and Shamir [19], Boneh *et al.* [23], and Cari *et al.* [40].

The generic attacks that were introduced in Sect. 1.3, indicate a general strategy that needs to be followed for white-boxing a cryptographic primitive.

- Key information needs to be spread over the entire implementation, forcing an adversary to analyze the whole implementation, instead of focussing on individual parts (that could for example be identified by an entropy study).
- Besides the key, all building blocks of the cryptographic primitive are publicly known (Kerckhoffs' principle). In order to prevent local analysis (introduction of break points or overwriting of components) similar to the key whitening attack (see Sect. 1.3.2), the building blocks should be randomized.

3.2.1 Initial White-Box Strategy

The strategy, proposed by Chow *et al.* [43] consists in transforming a given block ciphers into a randomized, key dependent network of lookup tables. This comprises three main steps.

1. Partial Evaluation. Embed the key into an operation, usually by transforming the (fixed) S-boxes S_i into key-dependent lookup tables T_i. E.g.,

$$T_i(x) := S_i(x \oplus k_i)\,,$$

in the case of a key addition operation before the S-box operation.

2. Tabularizing. This process consists in transforming all components of the block cipher, including the linear transformations, into lookup tables. This process may seem like 'black art' for those who are not familiar with white-boxing techniques. Also, there is no generic 'compiler' or algorithm for transforming a given algorithm into its tabularized equivalent. Instead, in the literature, a list of techniques is presented by means of demonstration on white-box implementations of the DES and the AES. We will introduce these techniques in Sect. 3.5 in a similar manner.

3. Randomization and delinearization. The reason why a transformation to lookup tables is used, is because lookup tables can implement any given function. Hence they are the ideal primitive to hide information. The main idea is as follows.

Observe a chain of three consecutive lookup tables in the network $L_3 \circ L_2 \circ L_1$, where L_2 contains some key information that needs to be hidden (e.g., $L_2(x) = x \oplus k$). Because the description of the lookup tables is available to a white-box

adversary, the key information can be extracted directly (evaluate $L_2(0)$). We prevent this by transforming the lookup tables as follows:

$$\begin{cases} L_1 & \rightarrow & L_1' = b_1 \circ L_1 \\ L_2 & \rightarrow & L_2' = b_2 \circ L_2 \circ b_1^{-1} \\ L_3 & \rightarrow & L_3' = L_3 \circ b_2^{-1}\,, \end{cases}$$

where b_1 and b_2 are bijections of dimensions corresponding respectively to the input and output dimension of the L_2 operation. L_i' are denoted as *encoded* lookup tables, and the encoded chain $L_3' \circ L_2' \circ L_1'$ has equivalent functionality to the original chain. The fundamental property is that L_2' does not leak any key information, and that an adversary is forced to analyze more components in order to gain information. In this case, an adversary would need to analyze L_1' and L_3' to recover information of b_1 and b_2 respectively. We briefly discuss these claims in Sect. 3.3.

Table 3.1. Implementation size of an n-bit to m-bit lookup table.

output size (bit)	input size (bit)			
	4	8	16	32
4	8 bytes	32 bytes	8 kB	512 MB
8	16 bytes	64 bytes	16 kB	1 GB
16	32 bytes	128 bytes	32 kB	2 GB
32	64 bytes	256 bytes	64 kB	4 GB

Unfortunately, lookup tables can be quite large to implement. To implement a n to m bit lookup table, $2^n \cdot m$ bits of storage is required. The exponential size in relation to its input is indicated by Table 3.1. Hence, for practical purposes, the input size is kept small, and a network of lookup tables is implemented instead of one big lookup table. The obfuscating internal encodings b_i cannot exceed the boundaries of the lookup table they encode, hence their effectiveness is reduced when reducing the lookup table dimensions. Therefore, before the tabularization process, other annihilating randomizations are injected, such as a randomly selected affine permutation (e.g., the *MB* operation in the white-box AES implementation).

This process of injecting annihilating bijections needs to be unpredictable, and difficult (ideally impossible) to be undone by an adversary.

3.2.2 Related Concepts

This strategy is similar to other protection techniques that appear in the literature. The method of randomization is for example a generic representation of

homomorphic functions. These are functions E that allow to compute on data in the encrypted domain (CED) [190], that is, to compute on data $E(x_i)$ instead of x_i. For example, the Paillier cryptosystem [144] is a homomorphic public key encryption scheme that enables the addition of x_1 and x_2 in the encryption domain, without the need to decrypt $E(x_1)$ or $E(x_2)$:

$$E(y) = E(x_1 + x_2) = E(x_1) \cdot E(x_2)\,.$$

Let us generalize this to an operation f on data x_1 and x_2 that we wish to compute in the encrypted domain. When f is described as a network of lookup tables, then the randomization strategy can be deployed as follows: generate random bijections b_1, b_2, b_3, and transform f into $f' = b_3 \circ f \circ (b_1^{-1} \parallel b_2^{-1})$. As a result, f' computes on the encrypted variables: $b_3(y) = f'(b_1(x_1), b_2(x_2))$, where the b_i refer to an analogue of encryption functions.

The ultimate goal in CED research is to find an efficient homomorphic encryption scheme for an algebraic ring, enabling both addition and multiplication onto encrypted values. Sadly, the best results in CED enable (infinite) additions, with only one multiplication [26]. This basic strategy has been deployed for hardening software. It modifies operations, randomizes data flow, and hence raises the bar against analysis. This was the germ for the initial white-box techniques proposed by Chow *et al.* [43, 42].

In 1982, Yao [196] presented a solution for the millionaires problem, where two millionaires want to know whom is the richest, without revealing their proper wealth. This led to the concept of *Secure Function Evaluation (SFE)* with seminal work of Goldreich, Micali, and Wigderson [73]. The core idea is that one party (Alice) creates a *garbled circuit*, and sends it to the other party (Bob). Bob is able to evaluate this circuit with his private inputs and compute the result. This process of evaluation requires that Bob receives the (garbled) values of Alice, without revealing any private information to each other. This is achieved via *Oblivious Transfer (OT)* [152]. Such a garbled circuit resembles a network of randomized lookup tables. The main difference however is that SFE is achieved by means of a two party protocol, while white-box cryptography aims to implement a stand alone functionality. Recent research in SFE aims to reduce the communication overhead [164, 111, 117], but unfortunately this leads to circuits of unpractical size for reasonable algorithms. However, both research fields (SFE and WBC) can inspire each other, and the ultimate goal would be a common solution.

In 1997, Patarin and Goubin [146] introduced a new asymmetric cryptosystem that is based on the difficulty of recovering two systems of multivariate polynomials from their composition. Their scheme introduce the idea of hiding one or

two rounds of small S-box computations with secret functions of degree one or two. The one-round schemes have shown to be insecure [146]. In [14], Biham cryptanalysed the two-round scheme (2R). However, this attack is not based on the decomposition of the polynomials. Dingfeng *et al.* [198] presented an attack based on the algebraic structure of the scheme with the intention of decomposing the compostion. The $2R^-$ schemes (where some polynomials describing the composition are kept secret), are not subject to any of these attacks. This basic idea was used by Carlier *et al.* [39] to obtain a way of representing block ciphers, concealing their design but leaving them executable.

Sander *et al.* [161, 119] built upon the Quadratic Residue Hypothesis [79] to hide polynomials and subsequently programs. Billet and Gilbert [20] proposed a new family of block ciphers, based on the Isomorphism of Polynomials (IP) problem [145, 147, 50], with traceability properties (see Sect. 5.5.1). The underlying building block was cryptanalysed by Faugère and Perret [64]. A repair for the traceable block cipher was proposed by Bringer *et al.* [31], based on a perturbation idea that was introduced to re-enforce the IP-based cryptosystems by Ding [60]. A similar approach was conceived in Bringer *et al.* [32], in an attempt to repair the white-box AES cryptanalysis, which we discuss below in Sect. 3.6.2.

3.2.3 Encoded Variants

Often, instead of implementing a cryptographic primitive E_k, an encoded variant is implemented. That is, a white-box implementation that is functionally equivalent to the encoded primitive $E'_k = G \circ E_k \circ F^{-1}$, where F and G are randomly selected bijections. There are two main reasons to follow this strategy.

1. To improve the security of the white-box implementation.

   The lookup tables of the white-box implementation are obfuscated using annihilating *internal* encodings b_i. The resulting lookup tables $L' = b_i \circ L \circ b_{i-1}^{-1}$ are difficult to analyze without the knowledge of the b_i encodings. However, the lookup tables at the beginning of the network are only protected by means of internal output encodings ($L' = b_0 \circ L$), and hence potentially more vulnerable to analysis. A similar reasoning holds for the final lookup tables. This could make white-box implementations vulnerable to chosen plaintext attacks (CPA), or chosen ciphertext attacks (CCA) when referring to the final round. To prevent analysis at the outer lookup tables of a white-box implementation, external encodings might be applied.

2. To prevent *code-lifting*.

   One of the major differences between a white-box model and the black-box/grey-box model is copyability. If cryptographic primitives are implemented in software, they can easily be duplicated. In one attack scenario

an adversary wants to extract the secret key, such that it can be used in a standard implementation. Instead, in a white-box model, an adversary could just use the implementation itself. E.g., isolate the decryption routine with embedded secret key from a DRM application, and use it in a stand-alone manner. This way, one can circumvent code for device or user authentication, no matter how robust the white-boxed decryption routine is obfuscated against key recovery. This is often denoted as *code-lifting*, and a related technique to prevent this attack, is called *Software Shielding* [180].

To prevent this, external encodings might be applied, which are embedded into the white-boxed cryptographic primitive. The annihilating encodings could be embedded into other parts of the application (e.g., in the user authentication code), or on other machines (e.g., in a remote execution scenario). We elaborate on the deployment of external encodings in practice in Chapter 5.

The use of encoded variants may be reasonable for DRM applications, or to prevent code-lifting. But they are not appropriate when standard encryption schemes are required. Also, they interfere with the deployment of a mode of operation, e.g., the CBC mode of operation, as depicted in Fig. 2.1. Hence, their deployment should be taken into consideration within the context of the application.

3.3 Security

3.3.1 Local Security

The core idea of white-box implementations, as proposed by Chow *et al.*, is to have *perfectly* obfuscated lookup tables. That is, lookup tables that do not leak any information. This is denoted as *local security*, and is similar to the concept of information theoretical security offered by the One-Time Pad.

Consider a lookup table L, that implements a key-dependent n-bit to m-bit function f_k:

$$\begin{array}{rccc} L: & \mathrm{GF}(2)^n & \rightarrow & \mathrm{GF}(2)^m \\ & x & & f_k(x) \end{array}$$

This lookup table is encoded using the randomly selected bijections b_0 and b_1, such that $L' = b_1 \circ L \circ b_0^{-1}$. When f_k is a bijective function, and b_0, b_1 are chosen uniformly from the full set of possible n-bit to n-bit and m-bit to m-bit bijections respectively, it can be proved information theoretically that L' does not leak any information about k. Intuitively: given the description of L' and $f_.$, then, with an equal probability for any k, there exists a b_0, b_1, such that $L' = b_1 \circ f_k \circ b_0^{-1}$.

However, *local security* does not apply in any other cases. When a lookup table L implements a lossy function, or the internal encodings are not taken at random from the full set of possible bijections, useful information may leak. E.g., in practice, often lookup tables of $2n$-bit to n-bit are implemented (e.g., to represent a XOR operation), where b_1 is taken from the set of bijective functions on $\mathrm{GF}(2)^n \times \mathrm{GF}(2)^n$ instead of $\mathrm{GF}(2)^{2n}$. We will demonstrate this vulnerability when describing the algebraic cryptanalysis in Sect. 3.6.2.

3.3.2 Metrics

Formalizing any other claim on the security of white-box implementations is difficult. For individual lookup tables, we can make some statements (such as local security). However, describing the security against analysis of a collection of lookup tables is hard. Furthermore, our goal is to implement block ciphers in a secure way, where most block ciphers do not even come with a (black-box) security description/proof themselves.

It is possible though, to set up some indicative metrics to account for the cryptographic quality of white-box implementations, by counting the number of different sets of lookup tables and the number of representations. A similar approach is followed to capture the 'security' of obfuscation techniques [45, 46]. Although they are not bullet proof security measurements, they are able to qualify the supposed robustness of white-box implementations, and provide a way to compare between the different techniques.

The following two metrics have been formalized in [42, 43]:
Diversity relates to the number of distinct implementations that could be compiled from one given algorithm with instantiated key, that is, the number of implementations that are functionally equivalent. This depends on the number of possible annihilating bijections b and affine encodings that can be applied, and can be counted as the number of possible encoding steps:

$$k \Rightarrow \#L' \,.$$

There exist $2^n \prod_{i=0}^{n-1}(2^n - 2^i)$ bijective affine transformations on n bits, since there are $\prod_{i=0}^{n-1}(2^n - 2^i)$ non-singular $n \times n$ matrices [58], while there are $2^n!$ possible bijective lookup tables. Hence the diversity of a network of randomized lookup tables is vast, even for small dimensions.

Ambiguity relates to the number of candidate keys for a given white-box implementation, lookup table, or set of lookup tables. This is a more important metric, as it is a better way to reflect the difficulty for an adversary to gain information on the original component. It is a measure of the number of alternative

interpretations an implementation can have, amongst which an adversary must disambiguate:

$$L' \Rightarrow \#k\,.$$

Metrics are only appropriate when there is a lack of efficient cryptanalysis results or security proofs. Therefore, in the following sections, we will discuss the state of the art of white-box implementations and of cryptanalysis results.

3.4 History

White-Box Cryptography was introduced in 2002 by Chow *et al.* in their seminal papers [42, 43], and defined the white-box attack context. Although they were the first to use the '*white-box* –' terminology, similar attack models have been presented before, such as the malicious host model by Sander and Tschudin [162, 163] and Hohl [94]. While public research in this topic mainly gained interest in the last decade, industry (e.g., to protect Intellectual Property) and governments (e.g., to protect military software implementations for in case they fall into enemies hands [116]) have been investigating similar concepts.

Chow *et al.* presented practical white-box implementations for the DES [43] and the AES [42]. Both implementations have been fully broken. In 2002, Jacob *et al.* [101] presented a cryptanalysis of the naked variant, that is, a variant without external encodings. They describe a fault-injection attack using differential cryptanalysis. This attack was improved by Link and Neumann [118] in 2005: they presented a new (naked) variant of white-box DES implementations that was resistant against any known attack at that moment. In 2007, all existing white-box DES implementations were cryptanalysed by Wyseur *et al.* [191], and independently by Goubin *et al.* [82]. Both attacks are based on a truncated differential cryptanalysis. Goubin *et al.* presented an attack that analyzes the first rounds of the white-box DES implementations, while Wyseur *et al.* presented an attack that works on the internal information, independently of the external encodings that are applied. The cryptanalysis result can be extended to generic Feistel ciphers, based on a similar strategy.

In 2004, Billet *et al.* [21] presented a very effective cryptanalysis of white-box AES implementations. Their attack is an algebraic cryptanalysis that operates on specific chosen sets of lookup tables, to remove the non-linear part of the internal encodings. This result opens the door to mount an algebraic analysis on a stripped down version of the white-box AES implementation. The attack was further improved by Michiels *et al.* [135] and can be deployed on a generic class of white-box implementations.

We outline the most important aspects of these cryptanalysis results in Sect. 3.6, while the details of the cryptanalysis by Wyseur *et al.* are described in Sect. 3.8.

3.5 Constructions

In this section, we present the white-box implementations of the DES, and the AES, as proposed in [118], and [42] respectively. The goal is to introduce the white-box transformation techniques that are needed to elaborate on their cryptanalysis results.

3.5.1 White-Box DES Implementations

We describe the white-box DES implementation of Chow *et al.* [43], and indicate some of the modifications that were presented by Link and Neumann [118], to improve the efficiency and robustness of the implementation.

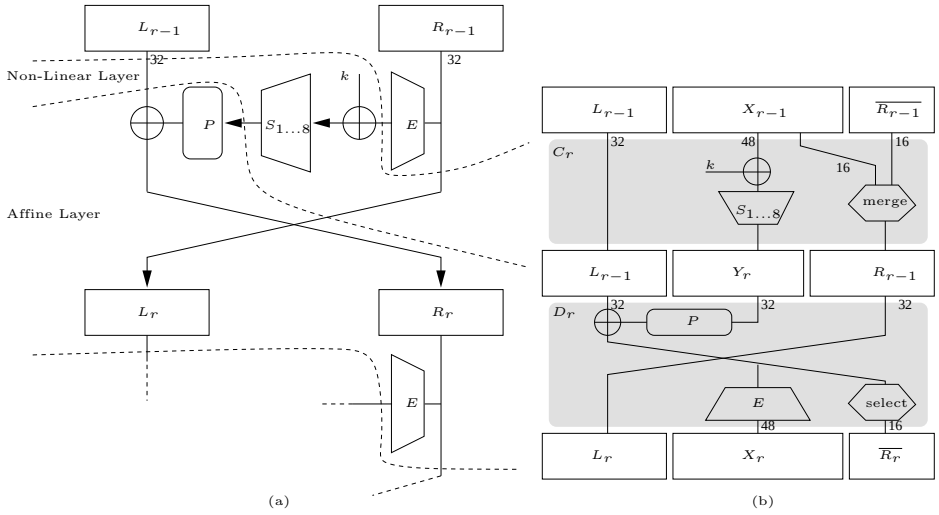

Figure 3.2. (a) One round of DES (b) One round of white-box DES

The DES is a Feistel cipher with 16 rounds, embedded between an initial permutation IP before the rounds, and its inverse permutation IP^{-1} after the

last round. Fig. 3.2 (a) depicts one round of DES. It has the following building blocks: an expansion operation E; an addition of a 48-bit round key K^r which is generated from the key schedule; 8 S-box operations S_i (each S-box is a non-linear mapping from 6 bits to 4 bits); and a bit permutation P. We refer to Sect. 2.2.1 for a detailed description of the DES.

The main idea of compiling DES as a white-box implementation is to express DES as an alternating layer of parallel lookup tables (C_r), comprising the non-linear part of the DES, and an affine transformation (AT) layer (D_r). These ATs can than be broken up (*matrix decomposition*) into smaller transformations, that can be described as lookup tables. Fig. 3.2 (a) depicts how the DES can be cut into two suitable layers; the layered representation is depicted in Fig. 3.2 (b). Notice that, following this strategy, it is thus possible to untie the Feistel structure of DES and implement the cipher as a substitution-permutation network (SPN) block cipher, as is the case with the AES.

T-boxes

The non-linear layer of each white-box DES round can be implemented as 12 parallel T-boxes, which are defined as follows:

$$\begin{cases} T_j^r = b_0 b_1 \,||\, b_2 b_7 \,||\, S_j(b_2 b_3 b_4 b_5 b_6 b_7 \oplus k_j^r) & (a) \quad \forall j = 1 \ldots 8 \\ T_j^r = b_0 b_1 b_2 b_3 \,||\, b_4 b_5 b_6 b_7 & (b) \quad \forall j = 9 \ldots 12 \end{cases}$$

Here r denotes the round number ($1 \leq r \leq 16$), $b_{0\ldots7}$ represent the eight input bits to each T-box, and k_j^r represents six bits of the round key. Fig. 3.3 depicts both types of T-boxes.

These 12 T-boxes represent the function C_r of round r of the DES implementation. The first eight T-boxes are called *non-linear T-boxes*, as they contain the non-linear S-boxes. Furthermore, each of these eight non-linear T-boxes bypasses two bits of L_{r-1}, and the two outer input bits to the S-box input (before key addition). Due to the construction of the DES S-boxes, which have a bijective relation between the four middle input bits $b_3 b_4 b_5 b_6$ and the output bits, these T-boxes are 8-to-8 bit bijections. This property is often referred to as *having entropy eight*. This design property is needed to prevent the T-boxes to leak information as described by Chow *et al.* [43]. Variations on the implementations [118, 192] can be introduced, such as re-arranging bypass bits, as long as the design properties of DES remain fulfilled.

The eight non-linear T-boxes bypass 16 bits of L_{r-1} and 16 bits of R_{r-1} in total. The other 16 bits of L_{r-1} and 16 bits of R_{r-1} are bypassed using four linear T-boxes. We call these $T_j^r (\forall j = 9 \ldots 12)$ *bypass T-boxes*, and the 16 bits of R_{r-1} they bypass we denote *restricted bits*, denoted by $\overline{R_{r-1}}$.

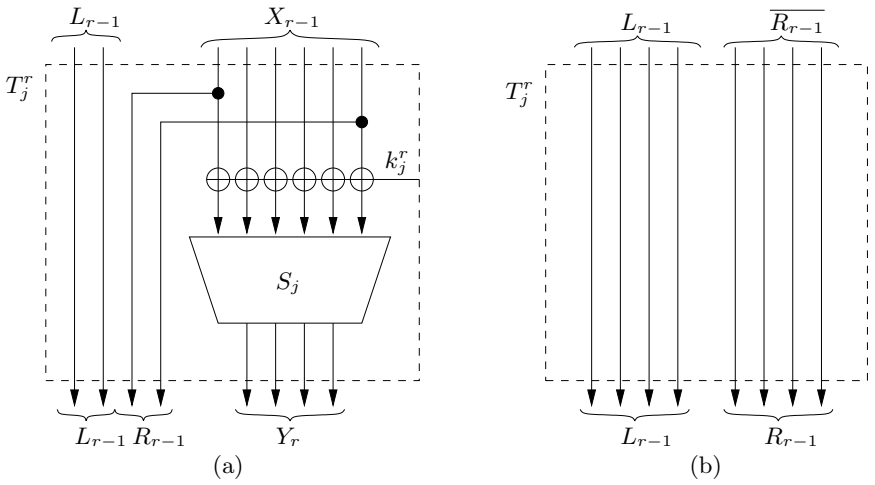

Figure 3.3. The two types of white-box DES T-boxes: (a) non-linear T-box with internal S-box (b) bypass T-box

A first randomization we apply is to shuffle the 12 T-boxes of C_r such that it is not straightforward to know which S-box is contained in which T-box. Shuffling is a linear operation, hence we can implement this operation in the affine operations D_{r-1} (shuffle) and D_r (unshuffle). Denote with π the shuffle operation which maps the T-box with internal S-box S_i onto $T_{\pi(i)}$.

Implementing the Linear Operations

The transformation D_r is a linear transformation of 96 bits to 96 bits. Denote with M the 96×96 bit matrix that represents this transformation ($D_r : y = Mx$). In order to implement this matrix operation as a network of lookup tables, we use a technique called *matrix decomposition*. The main strategy is to subdivide the 96×96 bit matrix M in $m \times n$ sub-matrices. Each $m \times n$ sub-matrix represents how an n-bit sub-vector of the input vector affects an m-bit sub-vector of the output vector. We implement each sub-matrix as a lookup table with 2^n rows of m bits.

In order to implement the full binary linear operation, the results of the outputs of the sub-matrix lookup tables must be XOR-ed. Each XOR operation applies to $2m$ bits and results into m bits. Putting it all together, implementing these XOR operations as a lookup table of 2^{2m} rows of m bits results into the full linear operation to be a network of lookup tables. Such a network is presented

in Fig. 3.4.

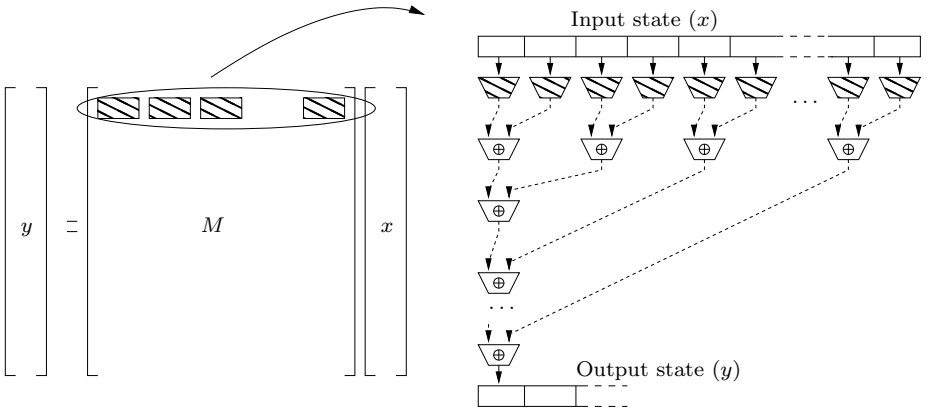

Figure 3.4. Matrix decomposition

Although lookup tables occupy much more space than linear operations, we need to implement the linear operations as lookup tables to apply non-linear encodings to the network. For both security and optimization purposes, D_r is decomposed into 8×4 lookup tables ($m = 4, n = 8$). Because every 8 bit T-box result of C_r is then preceding 24 ($= 96/4$) decomposed tables of the affine operation D_r, we can fold this T-box with every of its proceeding 8×4 lookup tables, and thus eliminate the 8×8 lookup tables, condensing the implementation. This optimization step has been suggested by Link *et al.* [118].

Re-Indexing and Delinearization

The steps mentioned above transform the DES into a network of lookup tables with an embedded key. At this point, these lookup tables do not provide much security as it is not difficult to extract the key from the content of the tables. To protect this content, input/output encodings are applied to these lookup tables.

Let L_i be a lookup table, and f_1 and f_2 be random bijections. Then $f_2 \circ L \circ f_1^{-1}$ is defined as its encoded version. We encode all lookup tables in the network, in such a way that the output encoding is cancelled by the input decoding incorporated into the next lookup table. Thus, consider a sequence of connected lookup tables

$$L := L_r \circ L_{r-1} \circ \ldots \circ L_2 \circ L_1 \,.$$

When we encoded each of these lookup tables, we obtain the chain

$$(f_r \circ L_r \circ f_{r-1}^{-1}) \circ (f_{r-1} \circ L_{r-1} \circ f_{r-2}^{-1}) \circ \ldots \circ (f_2 \circ L_2 \circ f_1^{-1}) \circ (f_1 \circ L_1 \circ f_0^{-1}),$$

which is functionally equivalent to the encoded chain $f_r \circ L \circ f_0^{-1}$. The encodings f_r and f_0 are discarded when a naked variant is desired. For practical implementation purposes (see Table 3.1), L might represent a network of lookup tables, and then f_i will be a cascade of non-linear encodings: $f_i = f_1^i \parallel f_2^i \parallel \ldots \parallel f_x^i$. This causes a loss in the possible diversity, which can partly be compensate for by introducing wide affine encodings to L (e.g., a re-arrangement of the bits in the T-box definitions, or shuffling of the order of T-boxes).

Result

As a result from the steps described above, we now have obtained a randomized, key dependent network of lookup tables that is functionally equivalent to DES, and for which it is hoped that is more difficult to 'break' in a white-box attack context, than a standard DES implementation.

An adversary in the white-box attack context has access to the definitions of the encoded lookup tables $L_i' = f_i \circ L_i \circ f_{i-1}^{-1}$, and any intermediate values during execution. For example, to the input to the white-boxed DES round r, which is defined as

$$f_1^{r-1}(v_1^r) \parallel f_2^{r-1}(v_2^r) \parallel \ldots \parallel f_{12}^{r-1}(v_{12}^r),$$

where v_i^r is the input to the (naked) T-box T_i^r. We define the inputs to the T-boxes using σ_r, where $(v_1^r \parallel v_2^r \parallel \ldots \parallel v_{12}^r) = \sigma_r(L_r, R_r)$. This captures any re-arrangement and duplication of the input bits.

Recommended Variant

The recommended variant as presented by Chow *et al.* implements the encoded DES cipher

$$\mathrm{DES}_k \rightarrow G \circ \mathrm{DES}_k \circ F,$$

where $F = M_1 \circ M_0$ and $G = M_4 \circ M_3$, while M_0 and M_4 are affine mixing bijections. A mixing bijection is a bijective affine transformation which attempts to maximize the dependency of each output bit on all input bits. M_1 combines the initial DES permutation IP and the expansion E, while M_3 combines the final operations such as the DES permutation P and the inverse initial permutation IP^{-1}. On top of that, both F and G are encoded by a cascade of nibble encodings in order to make them non-linear.

Without these external encodings, a white-box implementation (that is, a naked white-box implementation) becomes potentially vulnerable in the first and

last round. Attacks on naked implementations have been presented in [43, 101, 118].

Implementation Size

The original white-box DES implementation as presented by Chow *et al.* has an implementation size of 4.5 MB. A slight modification, as presented by Link and Neumann [118], reduces the implementation size to 2.3 MB. The main difference being the matrix decomposition into sub-matrices of size 8×4 (as described above) instead of 4×4, as presented by Chow *et al.*.

3.5.2 White-Box AES Implementations

We describe the white-box AES implementation proposal of Chow *et al.* [42]. As mentioned before, the general strategy is to merge several steps of the cipher into a network of lookup tables, which are then obfuscated using (random) input-output encodings. We present the white-boxing process for AES-128, which counts 10 rounds. The strategy is similar for AES-192, AES-256, and any other Rijndael implementation.

Construction

Denote by r the round number, and (i, j) the row and column index number of a byte in the state array.

First, the **partial evaluation** technique is deployed to merge the key addition with the subsequent S-box evaluation. The S-box function operating on the bytes during the `SubBytes` step is denoted by S. Then, we define a *T-box* as follows:

$$\begin{aligned} T^r_{i,j}(x) &:= S(x \oplus k^r_{i,j}) && \text{for } 1 \leq r \leq 9, \\ T^{10}_{i,j} &:= S(x \oplus k^{10}_{i,j}) \oplus k^{11}_{i,j-i}\,. \end{aligned}$$

Each output of the T-box plus the `ShiftRow` step contributes to 4 bytes of the state array after the `MixColumns` step. The contribution can be described by a 32×8 submatrix MC_i of the 32×32 bit matrix MC representing `MixColumns`. The entire function can be described as a 32×8 bit lookup table. The strategy is similar to the *matrix decomposition* operation presented in the construction of the white-box DES implementations above, which involves lookup tables that implement the addition operation (denoted by Chow *et al.* [42] as type IV tables, as depicted in Fig. 3.6.

This lookup table needs to be obfuscated with 4-bit nibble encodings.[3] To

[3] 4-bit nibble encodings were chosen instead of 8-bit encodings (which would obfuscate the full width of the T-boxes), for implementation size purposes. This allows the use of 8×4 addition operations, instead of 16×8, which would be 256 times larger to implement.

add to the diffusion, 8×8 affine *mixing* bijections are inserted before $T^r_{i,j}$ and a 32×32 affine bijections MB is inserted after the `MixColumns` part. The mixing bijection is canceled out by the preceding MC (round $r-1$), which includes the inverse (by simple matrix multiplication). Hence the inversion step is diffused over several lookup tables, making it hard to remove it. The resulting lookup table is depicted in Fig. 3.5.

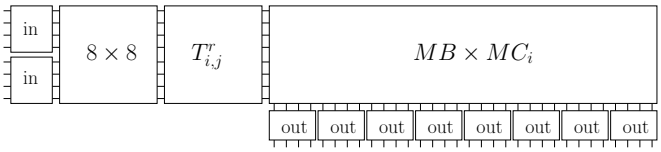

Figure 3.5. Type II table

To cancel the effect of the MB mixing bijection, a type III *untwist* table is implemented to take care of the inversion. The resulting lookup table is depicted in Fig. 3.6.

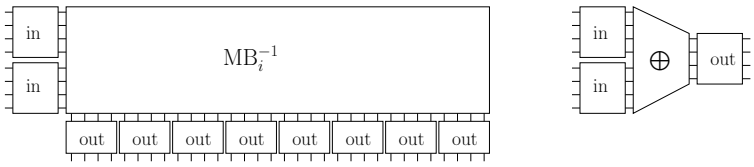

Figure 3.6. Type III and type IV table

To ensure that the encoded components of the type II and type III tables carry maximum information and to maximize the diffusion, we choose MB as a non-singular matrix where the 4×4 submitrices have full rank. Xiao and Zhou [194] presented an efficient method to generate such matrices.

External Input and Output Encodings

Finally, the external input and output encodings are implemented. Chow *et al.* [42] suggested to insert a 128×128 mixing bijection prior to the first T-box computation, and a 128×128 mixing bijection after the last T-box computation. The input encoding is composed with the inverted 8×8 input mixing bijection for T^1. These can be implemented using two sets of sixteen 8-bit lookup tables, depicted in Fig. 3.7, and a set of 960 addition tables (Type IV). Thus, instead of AES_k, $N_O \circ M_G \circ \mathrm{AES}_k \circ M_F \circ N_I$ is implemented, where M_F and M_G are the affine input and output encodings respectively, and N_I and N_O are the input and output non-linear nibble encodings respectively.

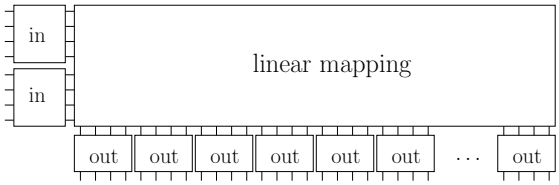

Figure 3.7. Type I table

Each *external input encoding table* represents the linear mapping associated with one 128×8 column of a 128×128 matrix – the composition of M_F and the concatenation of the input mixing bijections for $T^1_{i,j}$'s inverses – surrounded by 4-bit to 4-bit non-linear nibble encodings.

Each *external output encoding table* represents one 128×8 column of a 128×128 matrix – the composition of the output mixing bijection of the last round, one of the mappings $T^{10}_{i,j}$, and a column of the matrix M_G – surrounded by 4-bit to 4-bit non-linear nibble encodings.

Both the outputs of the input and the output encoding network need to be associated with a network of (encoded) addition tables to complete the implementation. These decode the outputs, xor together, and re-encode with the nibble encodings corresponding to the tables of type II and IV.

Implementation Size and Metrics

In Table 3.2, an overview is presented of the number of lookup tables that constitute the white-box implementation.

Table 3.2. White-box AES implementation size

Type	# lookup tables	# size
Type I	32	131072 bytes
Type II	144	147456 bytes
Type III	144	147456 bytes
Type IV	2688	344064 bytes
Total	3008	770048 bytes

In total, the white-box AES implementation as described above, with the external input and output encodings consists of 3008 lookup tables, and has an implementation size of 770 kbytes. In comparison, a standard AES imple-

mentation requires around 4 kbytes of implementations size and 300 operations (both lookup tables and XOR operations). While AES-CTR-128 (AES-128 in the counter mode of operation) has a benchmark performance of 15.98 cycles/byte on a Pentium 4 [150], white-box AES implementations in a similar mode has an estimated performance of 188 cycles/byte, almost 12 times slower. The slow-down is mainly caused by the implementation of addition operations into 2688 lookup tables. Although this seems a substantial price to pay in performance and memory size, for some applications it might be an acceptable trade-off.

Table 3.3. White-box AES implementation diversity and ambiguity

Type	Diversity	Ambiguity
Type I	$2^{1483.5}$	$2^{937.4}$
Type II	$2^{556.6}$	$2^{439.6}$
Type III	$2^{486.4}$	$2^{369.4}$
Type IV	$2^{132.8}$	$2^{84.6}$

Table 3.3 refers to the number of different interpretations a lookup table can have. Computing the ambiguity of groups of lookup tables, or even for the entire white-box implementation is rather hard, as it depends on relations that can be constructed between lookup tables.

3.6 White-Box Cryptanalysis

In this section, we present an overview of cryptanalysis results on white-box implementations. The techniques that are deployed are inspired by cryptanalysis techniques as introduced in Sect. 2.3 and Sect. 2.3.2. The method of deployment will be slightly different though, due to the open attack model. Particularly, intermediate information can be exploited and faults can be injected between steps in the implementation (at run time).

We distinguish two main classes of attacks. For clarity, we will present the details of our cryptanalysis of white-box DES implementations in a separate section (Sect. 3.8).

3.6.1 Differential Cryptanalysis

Differential attacks exploit the relation between differences of plaintext pairs, and the differences of the corresponding ciphertext pairs (see Sect. 2.3). While in black-box cryptography, a statistical comparison between input and output

difference is made, the white-box attack model allows a finer granularity. Namely, an adversary can observe any intermediate information between the building blocks[4] of the white-box implementation at execution time. This includes the input to the obfuscated rounds.

Differential cryptanalysis techniques have been mounted on white-box implementations in recent years [43, 101, 118, 82, 191], in particular on the white-box DES implementations.

In their seminal paper, Chow *et al.* [43] already admitted a vulnerability of their *naked* white-box DES implementation. They presented a *statistical bucketing attack*, which is similar to DPA attacks: guess key bits, and use differences (in DPA, statistical differences of power profiles) to confirm or deny the guess. The strategy described by Chow *et al.* is depicted in Fig. 3.8, and works as follows:

- Select an S-box S_i^1 of the first round. Because the implementation is naked, it is straightforward to find out which T-box implements S-box S_i^1. This can be achieved by selecting plaintext differences that affect S_i^1, and observe which T-box is affected. We denote a building block to be *affected* (by some difference Δ), when its input changes (due to the effect of this difference).
- Guess 6 bits of the round sub-key that affect S_i^1, and compute 64 plaintexts P, each corresponding to a different input to the S-box.
- Pick b, an output bit of S_i^1, and group the plaintexts into two sets I_0, and I_1 using a reference DES implementation and the key guess. $P \in I_0$ if $b = 0$, while $P \in I_1$ if $b = 1$.
- Select a T-box T_z^2 that implements an S-box S_j^2 that has bit b as input. Compute the set I'_0 of inputs to T_z^2 corresponding to the plaintexts $P \in I_0$. Similarly, compute the set I'_1. If our key guess is correct, then I'_0 and I'_1 must be disjoint sets, because they correspond to obfuscated inputs of T_z^2 that are different in at least bit b.

Iterated verification of key guesses for all S-boxes in the first round, eventually reveals 48 bits of the 56-bit key. The remaining 8 bits can be determined by exhaustive search. This cryptanalysis exploits the non-linearity of the S-boxes.

Link and Neumann [118] improved the statistical bucketing attack. They observed the 4-bit nibble encoded outputs of the first round of T-boxes T^1, corresponding to the output of S_i^1, to confirm or deny their key guesses, instead of a verification based on one bit at the input of the second round, thus greatly

[4] In the white-box implementations, presented by Chow *et al.* [43, 42], and Link and Neumann [118], all building blocks of the implementation are lookup tables.

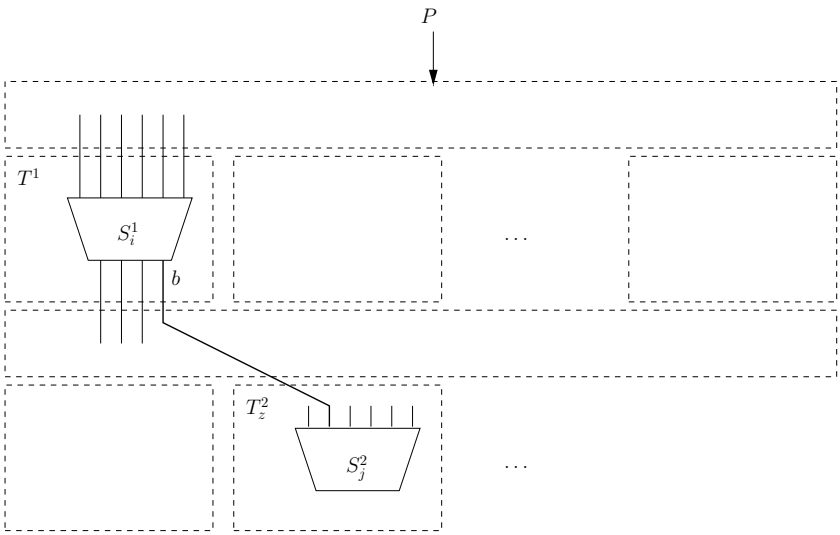

Figure 3.8. Statistical bucketing attack on naked white-box DES implementations

improving the efficiency of the statistical bucketing attack. Their attack relies on the fact that the 8-bit output of the T-boxes are protected by two cascaded 4-bit nibble encodings.

In order to prevent an attack on the split T-box output, Link and Neumann suggest to discourage analysis of the T-box output by eliminating it as an accessible intermedia variable. This can be achieved by decomposing the matrix preceeding the T-boxes into 8×4 blocks. Consequently, the 8×8 T-boxes and the corresponding 8×4 multiplication tables can be merged into a single 8×4 lookup table. To prevent the original statistical bucketing attack, an extra diffusion matrix δ is introduced, that mixes L_r bits with replicated bits of R_r. This defeats the detection of wrong key guesses, and hence successful deployment of the original statistical bucketing attack.

Fault Injection

In [101], Jacob *et al.* presented a differential fault analysis attack [18] on *naked* white-box DES implementations. The fault injection attack exploits the capabilities of an adversary in the white-box context to alter the implementation, or to modify the data flow at execution time. The attack strategy is as follows:

- Observe the final round of the implementation. Its input is C_{r-1}, a 96-

bit vector, the input to the 4-bit nibble encoded T-boxes. The output is the naked vector $C = (L_r, R_r)$. The attack computes the pre-image C_{r-1} of the final round for some specific ciphertext C. Note that, in order to compute this pre-image, a decryption routine corresponding to the white-boxed encryption scheme needs to be available (black-box access suffices).

$$C_{r-1} = \mathrm{WBE}_k^{1\ldots r-1} \circ D_k(C) = f_{r-1}^{-1}(\sigma_{r-1}(L_{r-1}, R_{r-1}))\,.$$

The pre-images C_{r-1} that Jacob *et al.* compute, are listed in Table 3.4, where i runs through the values $1 \ldots 32$, and F_r^k denotes the round key k_r addition, S-box operation and permutation operation.

Table 3.4. Pre-image computations for fault injection attack

$C = (L_r, R_r)$	(L_{r-1}, R_{r-1})
$(0, 0)$	$(F_r^k(0), 0)$
$(2^i, 0)$	$(F_r^k(0) \oplus 2^i, 0)$
$(0, 2^i)$	$(F_r^k(2^i), 2^i)$

- The fault injection takes place at the input of the final round, to reset the input from $(F_r^k(2^i), 2^i)$ to $(F_r^k(0), 2^i)$, where 2^i is the string that represents the integer 2^i. This can be achieved with overwhelming probability, by observing the 4-bit nibble input encodings that are affected by the differences between the plaintexts computed in Table 3.4. The outcome will be $C = (F_r^k(2^i) \oplus F_r^k(0), 2^i)$, from which the last round sub-key can be computed (due to the non-linear behavior of the S-boxes).

The attack by Jacob *et al.* is very efficient, and requires only very few operations. However, it requires the availability of a corresponding decryption functionality, and can only be deployed on the naked variant. Moreover, for the fault injection to succeed, the encodings must have the property that an attempt to modify L_{r-1} to $F_r^k(0)$ does not affect any bits of R_{r-1}.

Link and Neumann [118] improved the white-box DES implementation, to reduce the fault injection success probability, and strengthen the intermediate values against tampering. They proposed to implement the final addition tables as 16×8 lookup tables, such that the T-boxes can be encoded with 8×8 input encodings instead of nibble encodings. This gives the attack by Jacob *et al.* only a valid attack point with probability 0.25. Their measurements indicate that the resilience of white-box DES against the fault injection attack improves: on average 9.8 bits of the final round key can be learned rather than 40.

Cryptanalysis of Encoded Variant

In 2007, Goubin *et al.* [82] and Wyseur *et al.* [191] independently presented a differential cryptanalysis on white-box DES implementations.

Goubin *et al.* describe a *truncated differential cryptanalysis* on the naked and encoded variant of the white-box DES implementation as presented by Link and Neumann. The cryptanalysis is based on the *difference cancellation property* of the DES round (see Desmedt *et al.* [57]), to verify guesses on the round sub-key. We first explain the attack on the naked variant.

Observe the first round of the DES. (L_0, R_0) denotes the input to this round (the plaintext after IP). Then, the input to the next round (L_1, R_1) is defined as follows:

$$\begin{aligned} L_1 &= R_0 \\ R_1 &= L_0 \oplus f(R_0, k_r)\,, \end{aligned}$$

where f denotes the non-linear operation of the DES, which consists of the round key addition, parallel evaluation by 8 S-boxes, and the permutation P. The difference cancellation property states that Δ_{L_0} (the difference between L_0 and L'_0) can be chosen as such that $\Delta_{R_1} = 0$. Hence, for a difference Δ_{R_0}, we are able to compute Δ_{L_0} (given a correct key guess), such that only left bits to the input of the second round are flipped:

$$\begin{array}{lrcl} \text{given} & \Delta_{R_0} & = & R_0 \oplus R'_0\,, \\ \text{compute} & L'_0 & = & L_0 \oplus f(R_0, k) \oplus f(R'_0, k) \end{array} \quad \Rightarrow \quad \begin{array}{rcl} \Delta_{R_1} & = & 0 \\ \Delta_{L_1} & = & \Delta_{R_0}\,. \end{array}$$

As a result, we are able to control the flips of bits to the input of the second round. Verification of the key guess can occur by counting the number of affected T-boxes. Observe that, when Δ_{R_0} represents flips of bits with index 2 and/or 3 (b_4 or b_5 in our T-box definition), at most two T-boxes in the second round will be affected.

To mount this attack on the encoded variant, analysis of the input encoding is required. The approach consists in computing the images of a random vector X_0 at different levels of the obfuscated DES, and compare these values to the corresponding images of $X_0 \oplus \Delta$. This way, basis vectors for the matrix M_0^{-1} can be recovered, enabling a point of attack to deploy the naked variant strategy described above. We refer to Goubin *et al.* [82] for more details on this cryptanalysis.

Note however, that this cryptanalysis depends on the description of the external input encoding. It is not clear how nibble encodings on the external input encoding (as suggested by Chow *et al.*) affect the cryptanalysis. To thwart the cryptanalysis, non-linear external encodings could be applied, e.g., an external encoding that is similar to two obfuscated DES rounds with random key/S-boxes.

Cryptanalysis on the Internal Round Structure. In [191], we present a *truncated differential fault injection attack* on the internal rounds of the white-box DES implementation. We present a summary of the cryptanalysis here, and refer to Sect. 3.8 for a detailed description.

The attack described is deployed directly on the obfuscated round vectors $f_i^r(v_i^r)$; it analyzes the difference propagation caused by a fault injection on these vectors. Because on the one hand, the vectors $f_i^r(v_i^r)$ are encoded versions of the naked inputs inputs v_i^r of T-box T_i^r, and on the other hand the encodings f_i^r are non-linear, it is impossible to deduce which difference has been applied to the input of T_i^r. Therefore, the attack observes the difference propagation of Δ in the following rounds $(r+1, r+2, \dots)$.

The attack has the following steps:

- Observe the propagation of a difference Δ on the input of an encoded T-box T_i^r in rounds $(r+1, r+2)$, and select those differences that represent single bit flips on the inputs of the (obfuscated) S-boxes S_j^{r+1} by counting the number of affected T-boxes.
- Identify the non-linear T-boxes and by-pass T-boxes.
- As a result of the partial recovery of the non-linear encodings to the T-boxes T_z^{r+1}, and using specific properties of the DES S-boxes, the inputs to the S-boxes are (partially) recovered.
- Based on relations between inputs to S-boxes (on several rounds), recover the key bits (up to some natural ambiguity).
- In the case of *simple* external encodings, their definition can be recovered using the extracted secret key.

The cryptanalysis works on a subset of consecutive obfuscated rounds, regardless of the external encodings, and exploits specific properties of the DES round function. Through analysis of the difference propagation, information on the inputs to the S-boxes (after key addition) is gained, eventually leading to a point of attack to extract secret key information.

3.6.2 Algebraic Cryptanalysis

The strategy adversaries conceive in algebraic cryptanalysis is to model cryptographic primitives as a set of algebraic equations. Solving these equations eventually yields secret key information. Block cipher designers introduce a robustness against algebraic cryptanalysis by designing round functions that have a high algebraic degree. The combination of several rounds makes it infeasible

in practice to deploy algebraic cryptanalysis in a black-box context. In a white-box context, these attacks are in particular interesting, because more equations can be composed, that describe a smaller group of components of the implementation, e.g., a collection of lookup tables that describes an obfuscated round. The main idea can be described as follows: although none of the lookup tables (when considered individually) should leak sensitive information, the analysis of a composition of lookup tables could reveal information.

Stripping non-linear encodings from obfuscated rounds

Algebraic cryptanalysis of white-box implementations has been introduced by Billet *et al.* [21], with their cryptanalysis of white-box AES implementations.

They presented a novel way to remove the non-linear encodings of the obfuscated rounds. Recall that one obfuscated AES round consists of four mappings between four bytes of the input array to four bytes of the output array. Each mapping refers to one 4-bytes MixColumns operation, and is preceded by four parallel T-box $T^r_{i,j}$ operations which includes the round key addition. The mappings are (byte-)encoded with the input encodings $P^r_{i,j}$ and output encodings $Q^r_{i,j}$. Note that consecutive encodings are annihilating: $P^{r+1}_{i,j} = (Q^r_{i,j})^{-1}$. Denote by R^j_j such a mapping, which is depicted in Fig. 3.9.

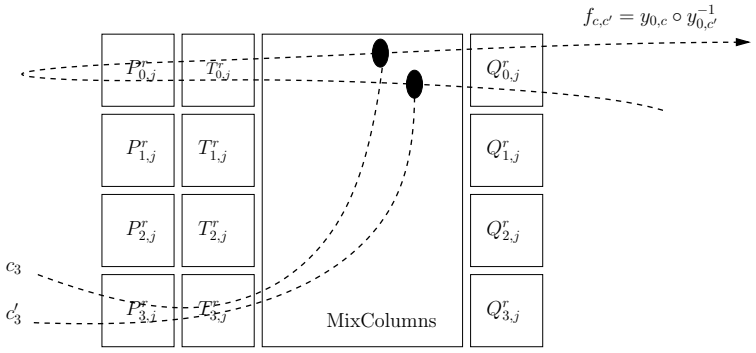

Figure 3.9. Analysis of an encoded AES round mapping

The main idea of the cryptanalysis is to observe the composition of lookup tables that correspond to a mapping R^r_j as one mapping of four input bytes (x_0, x_1, x_2, x_3) to four output bytes (y_0, y_1, y_2, y_3), and construct algebraic equations for this mapping.

Define $y_{0,c}(x)$ as the mapping from x_0 to y_0, where (x_1, x_2, x_3) are fixed to the constant $c = (c_1, c_2, c_3)$. Then,

$$y_{0,c}(x) = Q^r_{0,j}\left(\alpha T^r_{0,j}\left(P^r_{0,j}(x)\right) \oplus \beta_c\right) ,$$

where α and β_c originate from the affine mapping MC, and β_c depends on the constant inputs c. The mapping $y_{0,c}$ is bijective, due to the bijectiveness of $P^r_{0,j}$, $T^r_{0,j}$, $\mathrm{MC}_i|_{0..7}$, and $Q^r_{0,j}$. Hence we can define

$$f_{c,c'} = y_{0,c} \circ y^{-1}_{0,c'} = Q^r_{0,j}\left(\left(Q^r_{0,j}\right)^{-1} \oplus \beta_{c,c'}\right) ,$$

as depicted by the dotted line in Fig. 3.9, where $\beta_{c,c'} = \beta_c \oplus \beta'_c$. Note that $\beta_{c,c'}$ can take all values in $\mathrm{GF}(2^8)$ by varying the constant c' (even one c'_i will suffice). An adversary is able to compute this mapping $f_{c,c'}$, given input-output access to R^r_j. The mapping $y^{-1}_{0,c'}$ can be computed by computing a lookup table for $y_{0,c'}$ on all its 2^8 inputs, and inverting the obtained lookup table. The mapping $Q^r_{i,j}$ and $\beta_{c,c'}$ are unknown. Using Theorem 1, an adversary is able to recover the non-linear part of the encoding $Q^r_{0,j}$.

Theorem 1 *Given a set of functions $\mathcal{S} = \{Q \circ \oplus_\beta \circ Q^{-1}\}_{\beta \in \mathrm{GF}(2^8)}$ where Q is a permutation on $\mathrm{GF}(2^8)$, and $\oplus_\beta$ is the translation by $\beta \in \mathrm{GF}(2^8)$, one can construct a permutation $\tilde{Q}$ such that the mapping $A = \tilde{Q} \circ Q$ is affine.*

This was presented by Billet *et al.*, who described an efficient algorithm to compute such a permutation $\tilde{Q}$. As a result, an adversary is able to remove from the white-box AES round mapping R^r_j the non-linear part of the encodings $P^r_{i,j}, Q^r_{i,j}$ using the recovered permutations $(\tilde{Q}^{r-1}_{i,j})^{-1}$ and $\tilde{Q}^r_{i,j}$ respectively.

$$R'^r_j := \left\{ \Big\|_i \tilde{Q}^r_{i,j} \right\} \circ R^r_j \circ \left\{ \Big\|_i \left(\tilde{Q}^{r-1}_{i,j}\right)^{-1} \right\} ,$$

where $\tilde{Q}^{r-1}_{i,j}$ is obtained by analogous analysis of R^{r-1}_j. Remaining is a white-box AES round R'^r_j that is encoded by affine parasites of the form $A_Q = \tilde{Q} \circ Q$. A set of algebraic equations can be composed for R'^r_j from which, using specific design properties of the MixColumns operation, the affine parasites can be obtained. This eventually leads to the recovery of the AES-128 key. We refer to [21] for a detailed description of this cryptanalysis. The most important element to capture within the context of this thesis is the innovative strategy that is deployed to remove the non-linear part of the round encodings.

Generic Attack

Michiels *et al.* [135] have generalized the attack strategy described in Billet *et al.*, for a family of substitution-linear transformation (SLT) ciphers. The white-box techniques described by Chow et al. [43, 42] can be applied to derive white-box implementations of other block ciphers. The question remains for which block ciphers such a white-box implementation remains secure, given the recent cryptanalysis results. Therefore, generic attacks are in particular interesting. The attack strategy consists of three steps: (1) removal of the non-linear part of the round encodings, based on the process applied by Billet *et al.*, (2) removal of the linear part of the round encodings (based on solving an affine equivalence problem [22]), and (3) extraction of the secret key information by algebraic analysis of the stripped-down round function (by solving a matrix equivalence problem).

First, let us define the family of block ciphers on which this generic cryptanalysis applies.

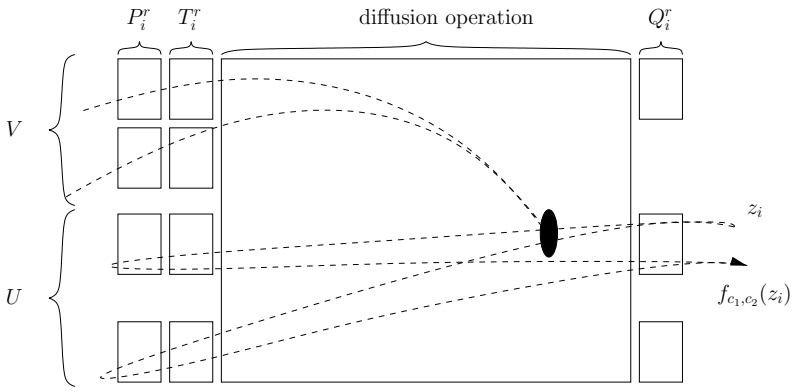

Figure 3.10. Generic analysis of SLT rounds

Definition 4 *(Substitution - Linear Transformation (SLT) ciphers). A cipher is defined as an n-bit* SLT cipher*, if it can be described as a number of rounds* $R^i, 1 \leq i \leq r$*, where each round is specified as 3 consecutive layers:*

- *a round key* k^r *addition,*
- *a layer of parallel S-boxes* $S_1^r, S_2^r, \ldots S_s^r$*. These S-boxes are (public) m-bit non-linear mappings, with* $n = m \cdot s$*, to realize confusion.*
- *a linear diffusion layer, which can be represented by an* $n \times n$ *matrix* M^r*.*

The operations S_i^r and M^r are part of the cipher specification (according to Kerckhoffs' principle).

The family of SLT ciphers, is a subset of the SPN ciphers, and includes ciphers such as the AES [138], Serpent [15], and KHAZAD [7].

The strategy of the cryptanalysis consists of the following steps.

1. Observe that Billet *et al.* construct a bijective relationship between x_0 and y_0 that can be influenced by the inputs x_1, x_2, x_3. Michiels *et al.*, generalized this to a bijective relationship between a subspace of U (input to P_i^r), and an output of Q_i^r, influenced by the choice of vectors of V. U and V are required to be disjoint sets.

   Similar to the approach in Billet *et al.*, a mapping f_{c_1,c_2} can be constructed, and thus a family $S = \{Q_i^r \circ \oplus_c \circ (Q_i^r)\}$ can be composed. If the mapping of vectors from V is surjective on the output of Q_i^r (i.e., surjective on $\mathrm{GF}(2^m)$), $c_2 \in \mathrm{GF}(2^m)$ can cycle through the entire space of its values. As a result, based on Theorem 1, the non-linear part of Q_i^r can be recovered.

   Now, an adversary can strip down the obfuscated white-box round function to an affine encoded round function R^r, thus having access to affine encoded inputs $a_i^r(v_i^r)$ to the T-boxes.

2. The next two steps of the cryptanalysis are intended to describe the affine encoded round function in a generic representation that can be further analyzed. First R^r is described as $\bigoplus_i T_i$, where T_i are m-bit to n-bit T-boxes. Remark that this only applies to ciphers with an affine permutation network. For example, this also applies to the Serpent family of ciphers.

   Next, the round function is described as a SAT cipher round function. That is, a cipher for which the round function is a cascade of S-boxes L_i^r, followed by an affine transformation b^r. These components L_i^r and b^r can be computed by an adversary.

   Note that at this point, the obfuscated cipher can be inverted, because the b^r operation is affine, invertable, while the S-boxes L_i^r can be inverted by inverting the (bijective) lookup table description.

3. The last step performs the round key extraction. This is achieved by obtaining the equivalence between the computed L_i^r, and the S-boxes S_i^r from the cipher description. An algorithm to compute an affine equivalence $L = C \cdot S \cdot D$ is described by Biryukov *et al.* [22]. The substitution box L_i^r includes the round sub-key k_i^r operation. The algebraic equations that

need to be solved are:

$$\begin{cases} L_i^r &= c_i^r \circ S_i \circ d_i^r \\ \{\|_i\ c_i\} &= b^{-1} \circ \{\|_i\ a_i^{r+1}\} \circ M \\ \{\|_i\ d_i\} &= \oplus_k \circ \{\|_i\ a_i^r\}^{-1}, \end{cases}$$

where c_i, d_i are the affine functions that describe the affine relation between L_i^r and S_i^r, and d_i^r contains the key addition operation $\oplus_k$. Solving these equations eventually leads to the secret key k, based on a linear equivalence solver for matrices.

To conclude, we note that the algebraic analysis of white-box implementations mainly constitutes of equivalence solving. Extracting the non-linear encodings from $Q \circ \oplus_\beta \circ Q^{-1}$ by Billet *et al.*, solving a linear equation based on the algorithm by Biryukov *et al.*, and recovery of secret key information based on a matrix equivalence solving by Michiels *et al.*

As described in related work, the traceable block cipher introduced by Billet *et al.* [20] can be regarded as a white-box implementation. This has been cryptanalyzed based on a equivalence solver for isomorphic polymorphisms by Faugère *et al.* [64].

The deployment of equivalence solvers for white-box implementations is a natural process, because the main idea of obfuscating block ciphers consists in constructing $F \circ L \circ G$, for random bijections F, G, in order to hide (key) information of L. It is of interest to perform further studies on equivalence solvers for various building blocks, in order to obtain mathematical structures that are suitable for white-box implementations.

Analysis of a Basic Building Block

The analysis, as introduced by Billet *et al.* to remove the round encodings from white-box AES implementations, and by Michiels *et al.* for generic white-box SLT implementations, can be applied to other white-box building blocks besides the round functions. Observe that in both the white-box DES implementations, and the white-box AES implementations, trees of XOR tables are implemented (referred to as type IV tables in the white-box AES implementation). Although implementing XOR operations as lookup tables imposes a negative effect on computation speed and implementation size, they are required in order to deploy non-linear encodings on the keyed building blocks. Moreover, de-linearizing affine transformation as trees of encoded lookup tables is the basic method to make implementations such as the white-box DES implementation robust against inverting.[5] However, we show that this strategy fails.

[5]Note however, that invertibility is not an general requirement. E.g., in a traitor tracing scheme or secure storage scenario, the PR-CPA notion is often not demanded.

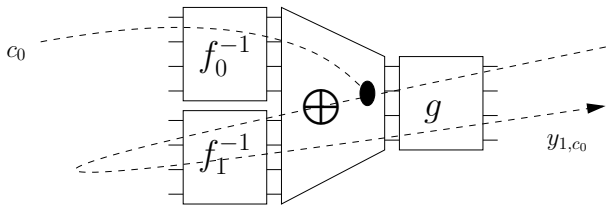

Figure 3.11. Analysis of an encoded XOR building block

In this dissertation, we show how the cryptanalysis strategy introduced by Billet *et al.* can be applied to XOR tables. The y_{1,c_0} function can be computed by an adversary that has input-output access to the encoded XOR building block:

$$y_{1,c_0} := g \circ \bigoplus_{f_0^{-1}(c_0)} \circ f_1^{-1} ,$$

where f_0, f_1 are the (yet unknown) input nibble encodings to the XOR building block, and g the output nibble encoding. Observe that for any input c_0, y_{1,c_0} is bijective. Hence, the family

$$y_{1,c_0} \circ y_{1,c_0'} = g \circ \bigoplus_{f_0^{-1}(c_0) \oplus f_0^{-1}(c_0')} \circ g^{-1} ,$$

can be computed. From this family of functions $\{g \circ \oplus_c \circ g^{-1}\}$, where c takes all possible values (which can easily be achieved by running over all possible c_0' because f_0^{-1} is surjective), the non-linear part $\tilde{g}$ of g can be derived, that is, $\tilde{g}$ such that $\tilde{g} \circ g = a_g$, with a_g an affine function.

As a result, the non-linear parts $\tilde{f}_0, \tilde{f}_1$ of the encodings f_0, f_1 respectively can be computed

$$\begin{cases} \tilde{f}_0 &= \tilde{g} \circ y_{0,c_1} \\ \tilde{f}_1 &= \tilde{g} \circ y_{1,c_0} . \end{cases}$$

Hence, we obtain a re-linearized XOR building block

$$a_g \circ \bigoplus \circ \{a_{f_0} \parallel a_{f_1}\} = \tilde{g} \circ \mathrm{BB} \circ \left\{ \tilde{f}_0^{-1} \parallel \tilde{f}_1^{-1} \right\} ,$$

where *BB* denotes the original (de-linearized) building block, and a_i are affine functions.

Applying this strategy to the final lookup tables of an encoded XOR tree, immediatly reveals the encodings of the entire tree. Observe that $\tilde{f}_0$ is the non-linear

component of the output encoding of the XOR table at the branch preceding f_0^{-1} in the XOR tree. Hence, an XOR tree can be re-linearized efficiently.

The security metrics described in Sect. 3.3.2 did already indicate the potential vulnerability of such building blocks. The XOR building block in the white-box AES implementation has by far the weakest ambiguity.

A similar analysis can be achieved for other basic building blocks that have a similar *double bijective* behavior such as group addition in GF(2^n). Hence, to construct secure white-box implementations, the security of the implementation should not rely on the robustness of the obfuscation of these building blocks, as they leak information. We will show that this is the case for the white-box implementations of AES and DES.

Defeating the Initial Obfuscation Strategy

For XOR tables at the edges of a tree, the re-linearization process reveals encodings of other building blocks. This results into a generic point of attack. Once white-box implementations have been (partially) linearized, they become a target for algebraic analysis. For example, the nibble encodings of the type II tables in the white-box AES implementation, can be re-linearized in this way, which leads to an alternative mounting of the attack by Billet *et al.* [21].

Similarly, an alternative attack on white-box DES implementations might be constructed. Analysis of the addition tables at the tail of the addition trees in round r leads to recovery of the non-linear part of the input encodings of the T-boxes in round $r+1$. The head of the addition tree consists of the sub-matrices M_{ij} of the matrix M (which represends D_r, see Sect. 3.5.1), who's output encodings can be linearized with the analysis technique described above. As a result, the adversary has obtained knowledge of the affine encoded T-boxes (merged with the sub-matrix M_{ij}):

$$a_{ij}^r \circ M_{ij} \circ T_j^r \circ a_j^r \,,$$

where $a_z = A_z \oplus b_z$ are the (unkown) affine encodings, A_z its linear function, and b_z a vector. Denote with $\overline{T}_j^r$ the T-box without the key addition: $T_j^r = \overline{T}_j^r \circ \bigoplus_{k_j^r}$. Since M_{ij} and $\overline{T}_j^r$ are known, we can deploy the affine equivalence solver by Biryukov *et al.* [22], and obtain a pair of affine functions $(e_j, e_{ij}) = (E_i \oplus d_i, E_{ij} \oplus d_{ij})$ such that

$$\bigoplus_{d_{ij}^r} \circ E_{ij}^r \circ M_{ij} \circ \overline{T}_j^r \circ \bigoplus_{d_j^r} \circ E_j^r = a_{ij}^r \circ M_{ij} \circ \overline{T}_j^r \circ \bigoplus_{k_j^r} \circ a_j^r \,.$$

Note that this pair (e_j, e_{ij}) is not unique. In order to recover the key information, we correlate the obtained information for T-box T_j^r with similar information for all the T-boxes T_z^{r-1} of round $r-1$. This gives a new point of attack, since the input to E_{jz}^{r-1} is equivalent to the input of D_{r-1} (see Fig. 3.2 (a)), up to an ambiguity in the pairs (e_j, e_{ij}). When this ambiguity can be solved, then,

$$e_j^r \circ \mathrm{XOR} \circ \Big\|_{i=1}^{12} e_{ij}^{r-1} = \bigoplus_{k_j^r} \circ D_{r-1}\,, \tag{3.1}$$

where XOR denotes a (known) network of affine encoded XOR operations, is equivalent at the input of the S-box of $\overline{T}_j^r$. Since $D_{r-1}(0) = 0$, evaluating Equation (3.1) in the value 0 yields k_j^r. However, it remains to be seen up to what extend the pairs (e_j, e_{ij}) can be disambigated.

In the case that mixing bijections are introduced, two addition networks are implemented (one corresponding to the matrix J, another to the matrix K), instead of the addition table corresponding to M, with $M = J \circ K$, and J, K have sub-matrices of full rank. Since the sub-matrics K_{ij} are unknown, we cannot deploy the attack as described above. However, we might include K_{ij} in the affine encodings, and recover this with the equivalence solver. The result is a more complicated attack, since also the J_{ij} have to be taken into account when solving the relations between the recovered affined encodings, but nevertheless make white-box DES implementations prone to this cryptanalysis technique.

We might also extend the XOR builing block attack to other lossy building blocks. However it remains to be investigated up to what extend information can be extracted when the 'double bijective behaviour' is not present. Also, it is essential that the underlying structure of the encoded building block is known (which is for example not the case when mixing bijections are introduced). Hence, the security of white-box implementations should not rely solely on the injection of random annihilating encodings.

3.7 Running White-box Implementations Backward

When Chow *et al.* introduced white-box cryptography, their main objective was to prevent key extraction from the white-box implementations. However, it should be clear that they also attempted to introduce some one-wayness. This is mainly the case for the white-box DES implementations. When we do not consider the cryptanalysis results (since key extraction makes DES trivially invertible), it would require an effort of $16 \cdot 2^{96}$ to invert the functionality (note

that a brute-force black-box attack to invert the functionality only required 2^{56} effort). This is mainly by virtue of the addition networks. Every addition network in white-box DES implementations consists of 24 layers of networks that map 96 bits onto 4 bits, which is difficult to invert. However, given the generic attack on the XOR building block, the addition tables become linear, which makes the addition network invertible.

The white-box AES implementations have an inherently weak asymmetric behavior. Inverting their functionality requires only $40 \cdot 2^{32}$ computational effort, since every 32-bit column (see Fig. 2.4) can be inverted seperately. This effort is much less than brute-force inverting an AES implementation (effort 2^{128}), hence a white-box adversary has a significant advantage to a black-box adversary). However, this is specific to white-box AES implementations. It is plausible that other white-box implementations offer a stronger resistant against inverting attacks.

Note that the generic attack by Michiels *et al.* [135] makes a white-box implementation of an SLT cipher invertible after the second step in their cryptanalysis.

3.8 Cryptanalysis of White-Box DES Implementations with External Encodings

In [191], we have presented an algorithm to extract the secret key from white-box DES implementations. We elaborate on the cryptanalysis in this section.

This cryptanalysis is based on the observation that the white-box implementations that have been proposed so far, fail to obfuscate the fault propagation of the cipher they intend to obfuscate. Unbalanced fault propagation of the block cipher might be detected in its white-box implementations. Observe that the DES cipher has an unbalanced fault propagation, because of the Feistel structure, where only half of the bits are evaluated by non-linear S-boxes in one round.

In a white-box attack context, an adversary has full access to observe and manipulate intermediate data (even at run-time). As a result, an adversary has access to the inputs of the obfuscated round functions, denoted as $v^r = \|_i v_i^r$, where v_i^r are the encoded inputs to T-box T_i^r for round r. Denote with f_j^r the input encoding of T_i^r. We mount our cryptanalysis on white-box DES implementation as follows: for the input of some round, we inject a fault on v_i^r. The objective is to identify faults Δv_i^r that correspond to some specific difference on $f_i^r(v_i^r)$, e.g., a flip of a single bit on the input of the S-box S_i^r. However, because f_i^r is a non-linear bijective mapping, the corresponding difference $\Delta f_i^r(v_i^r)$ cannot be deduced immediately. Instead, we observe the difference propagation at the output of each round (i.e., the input of the next round), to identify what difference $\Delta f_i^r(v_i^r)$ is caused by the fault injection on v_i^r. With this knowledge, we mount a truncated differential analysis on the DES implementation, which allows us

to retrieve a representation of the secret key that has been embedded into the white-box DES implementation.

Because the cryptanalysis only requires the input-output relation of the obfuscated rounds, it is independent of round obfuscation techniques. Also, the cryptanalysis can be mounted on an arbitrary set of consecutive round functions, and there is no restriction which window of consecutive rounds is targeted. Empirical results show that eight consecutive rounds will satisfy. This is the main difference with the cryptanalysis by Goubin *et al.* [82]. Our cryptanalysis is independent of the external encodings.

We structure our cryptanalysis in five main steps, and present the details of every step in the remaining part of this section.

1. First, we distinguish a set of faults on the inputs to a obfuscated round r, that represent differences that propagate the slowest through the *modified*[6] DES rounds. With overwhelming probability, the obtained set of faults will represent differences that flip only (one or two) restricted bits (denoted as $\overline{R_{r-1}}$).

2. Based on this set of faults, we distinguish faults to the input of round $r+2$, that represent differences that flip only one of the right half of bits (denoted as R_{r+1}).

3. With knowledge of the relation between faults at the input of an obfuscated round, and the difference it represents to the underlying DES implementation, we can apply differential cryptanalysis. This leads to the identification of the S-boxes that are implemented by the T-boxes, and eventually to the (partial) recovery of the inputs to these S-boxes (after key addition).

4. Based on this information, we can compute a representation of the secret key that is used to implement the white-box DES implementation, up to some natural ambiguity.

5. As the inputs to the S-boxes is known, and we have information on the key, we are able to recover the shielding external encodings.

Let us elaborate on the details of the cryptanalysis below. The steps used in the summary above, corresponds to the steps in the different parts of the cryptanalysis below.

[6]Note that for the obfuscated DES, rounds are defined differently. That is, a DES round is defined to start (and end) just *after* the expansion operation; the Initial Permutation at the input incorporates the expansion operation of the first round, while the last round does not implement the expansion operation.

Initialization

Denote with the *internal state* before round r, the 96 bits that represent the encoded version of $L_{r-1}||X_{r-1}||\overline{R_{r-1}}$. This is a 12-bit vector $v_1^r||v_2^r||\dots||v_{12}^r$, where v_j^r stands for the encoded input to a T-box T_j^r. This is a vector, and attacker is able to observe and manipulate.

Proposition 1 *An adversary is able to distinguish, isolate, and compute with the obfuscated round functions WBDES*$^r : \mathrm{GF}(2^8)^{12} \rightarrow \mathrm{GF}(2^8)^{12}$.

It is reasonable to assume that Proposition 1 holds, since this reflects that we wish to avoid *security through obscurity*, in accordance to the Kerckhoffs principle. For white-box DES implementations, it is particularly easy to identify the boundaries between the obfuscated rounds, since the addition networks can be distinguished easily. Denote with WBDES^r $(1 \leq r \leq 16)$ an obfuscated round function of a white-box DES implementation. Then, for a fixed input X, we can compute all the internal states v_j^r $(1 \leq j \leq 12; 1 \leq r \leq 16)$, using these round functions:

$$(v_1^{r+1}, v_2^{r+1}, \dots, v_{12}^{r+1}) = \mathrm{WBDES}^r(v_1^r, v_2^r, \dots, v_{12}^r).$$

We store all the values v_j^r, corresponding to the input X, and perform our cryptanalysis based on a fault injection on these internal states, as we have mentioned in the cryptanalysis strategy above.

We define a difference Δv on the input of T_j^r, as the difference between the original input value v_j^r, and the perturbated value v', caused by a fault injection.

$$\Delta v = v_j^r \oplus v' \in \mathrm{GF}(2^8)\,.$$

In the white-box attack context, an adversary can always choose which fault he wishes to inject. This in contrast to side-channel cryptanalysis, where it is difficult to inject a fault, such that the perturbated result is a pre-specified value. This is an important difference between a white-box model and a grey box model, and is crucial to deploy a cryptanalysis to white-box implementations.

3.8.1 Finding Restricted Bit Flips

Let T_j^r be an arbitrary encoded T-box in round r, encoded with input encoding f_j^r. In the first step of this cryptanalysis, we present an algorithm to construct the set $\mathcal{S}_{\overline{R}}(T_j^r)$

$$\mathcal{S}_{\overline{R}}(T_j^r) = \{\Delta v = v_j^r \oplus v' \mid v' \in GF(2)^8; v' \neq v_j^r; f_j^r(v_j^r) \oplus f_j^r(v') \text{ an } \overline{R_{r-1}} \text{ bit flip}\},$$

of input differences to the encoded T_j^r which represent flips of *one or two* restricted bits ($|f_j^r(v_j^r) \oplus f_j^r(v')| = 1, 2$).

The algorithm consists of two parts: (1) constructing the set $\mathcal{S}_R(T_j^r)$ of all differences which represent flips of *one or two* bits of R_{r-1}, and (2) to divide this set into $\mathcal{S}_{\overline{R}}(T_j^r)$ and $\mathcal{S}_{R\setminus\overline{R}}(T_j^r)$.

Finding Single R_{r-1} Bit Flips

Let $\Delta v : v_j^r \rightarrow v_j^r \oplus \Delta v$ be a difference of the input of T_j^r while the inputs v_l^r to the other T-boxes T_l^r are fixed to the values from the initialization phase ($\forall l \neq j : \Delta v_l^r = 0$). The following two properties can be proved for Δv.

Property 1 *If Δv represents a single bit flip of R_{r-1}, then in round $r+2$, at most two T-boxes are affected.*

Proof: When Δv represents a flip of a single bit of R_{r-1}, then in round $r+1$ it represents a flip of a single bit of L_r, as the reader can deduce from Fig. 3.2 (b). Because of the expansion and selection operation, this will result into two bits flipped to round $r+2$ (one of X_{r+1} and one of $\overline{R_{r+1}}$; or both X_{r+1} flips). Thus at most two T-boxes in round $r+2$ are affected. □

Remark that for some white-box DES implementations, a flip of *two* R_{r-1} bits has the same behavior, when both touch the same S-box in round $r+2$ (these bits will then be the two middle bits). Therefor they cannot be distinguished from single bit flips.

In the white-box DES implementations presented by Chow *et al.* and Link *et al.*, eight such double flips will occur for each round. Depending on the definition of σ_r, the number of double flips can vary to less than 8, but this does not influence our cryptanalysis. To keep the discussion clear, in the remainder of the cryptanalysis, we assume the by-pass bits are ordered according to the design by Chow *et al.*, and therefore eight double bit flips will satisfy property 1.

Property 2 *When Δv is a difference such that a bit of L_{r-1} or Y_r is flipped, then in almost all cases more than 2 T-boxes are affected in round $r+2$. The exceptions (false positives) can be detected by repeating this process up to α times with different fixed inputs to the other T-boxes T_l^r.*

Proof: In round $r+1$, besides bypass bits, these differences represent flips to the inputs of S-boxes. Therefore, the number of flips to the inputs of round $r+2$ explodes, and strictly more than 2 T-boxes will be affected.

There are a few exceptions in which not more than 2 T-boxes are affected (*false positives*). Observe an affected S-box in round $r+1$. (There will always be at least one affected S-box). The input to this S-box changes in at least one and

at most 3 bits (one for Y_r and two for L_{r-1} bit flips). The effect on the output bits of this S-box depends on its other input bits, which depend on the inputs v_l^r set at the initialization phase. Hence the number of affected T-boxes in round $r+2$ will very likely change if we set other inputs to T_l^r, witch $l \neq j$. With a very high probability, two extra checks are sufficient to detect these false positives, if we change all the inputs to the other T-boxes ($\alpha = 2$). □

A faster selection can be implemented when taking the propagation to the input of round $r+1$ into account. For the differences described above, one single T-box will be affected, namely the T-box that by-passes the flipped bit.

These two properties give us a means to distinguish differences Δv that represent flips of R_{r-1} bits, and compose the set $\mathcal{S}_R(T_j^r)$. Algorithm 2 describes this procedure. The total number of differences representing flips of bits of R_{r-1} for all the T-boxes of one round, is exactly 40: 16 *single* flips of bits of R_{r-1} originating from X_{r-1}, 16 *single* flips of bits of $\overline{R_{r-1}}$, and 8 *double* flips of bits of $\overline{R_{r-1}}$.

Algorithm 2: Selecting obfuscated R_{r-1} bit flips on input of T_j^r

```
Set all v_l^r (1 ≤ l ≤ 12)
For all Δv ∈ GF(2)^8\{0}
    Compute 2 round functions with perturbated input v_j^r ⊕ Δv on the
    input of T_j^r
    if # affected T-boxes > 2 then
        break;
    end
    Perform extra checks, with v_l^r ←^R GF(2)^8_(1 ≤ l ≤ 12; l ≠ j)
    Δv → S_R(T_j^r)
end for
```

Split R_{r-1} into $\overline{R_{r-1}}$ and $R_{r-1}\backslash\overline{R_{r-1}}$ Flips

Let Δv represent flips of R_{r-1} bits. The following properties can be proved for $\Delta v \in \mathcal{S}_R(T_j^r)$.

Property 3 *If Δv represents a flip of bits of $\overline{R_{r-1}}$, there are exactly 2 propagated differences in round $r+2$: Δm, Δn. One (say Δm, input difference to T-box T_m^{r+2}) will affect strictly more than 2 T-boxes in round $r+4$, the other difference will affect at most 2 T-boxes in round $r+4$. Moreover, T_m^{r+2} will be a non-linear*

T-box; Δm represents flips of one or both of the two middle bits of the internal S-box; and Δn represents flips of respectively one or two $\overline{R_{r+1}}$ bits.

Proof: Let $\Delta v \in \mathcal{S}_R(T_j^r)$ represent a flip of single (or double) bits of $\overline{R_{r-1}}$. Then, in round $r+2$, this will propagate to a flip of one (or both) of the two middle input bits of an S-box S_m in T-box T_m^{r+2}. Hence T_m^{r+2} is a non-linear T-box. Denote Δm the propagated input difference to T_m^{r+2}. Furthermore, this flip will also be bypassed because of the selection operation (see Fig. 3.2 (b)). If this would be bypassed by T_m^{r+2} as well, then this T-box has an entropy of seven, in contradiction to the T-box design. Thus a second T-box T_n^{r+2} is affected, with input difference Δn. Therefor, Δv will affect exactly two T-boxes T_m^{r+2}, T_n^{r+2} with input differences $\Delta m, \Delta n$.

Consider the following DES S-box design properties [48]:

$$\Delta_{in} = 0wxyz0 \quad \Rightarrow \quad |\Delta_{out}| \geq 2 \tag{3.2}$$

$$|\Delta_{in}| = 1 \quad \Rightarrow \quad |\Delta_{out}| \geq 2\,, \tag{3.3}$$

with Δ_{in} the input difference to an S-box, Δ_{out} its resulting output difference, and $wxyz \in GF(2)^4 \backslash \{0\}$. Because of (3.2), Δm represents a flip of at least two Y_{r+2} bits at the output of the S-box. Due to the DES permutation P diffusion property and (3.3), Δm will affect more than 2 T-boxes in round $r+4$. Δn represents a flip of bits of R_{r+1}, and affects no more than two T-boxes in round $r+4$ (see Property 1). □

Property 4 *If Δv represents a flip of bits of $R_{r-1} \backslash \overline{R_{r-1}}$, there are exactly 2 propagated differences in round $r+2$. Both affected T-boxes are non-linear T-boxes, and each of their input differences will affect strictly more than two T-boxes in round $r+4$.*

Proof: If $\Delta v \in \mathcal{S}_R(T_j^r)$ represents a flip of bits of $R_{r-1} \backslash \overline{R_{r-1}}$, then for 2 S-boxes in round $r+2$, exactly one input bit will be affected, and thus exactly 2 non-linear T-boxes in round $r+2$ are affected.

Because of S-box design property (3.3), each of these differences will represent a flip of at least two Y_{r+2} bits. As a consequence of the DES permutation P diffusion property, both these differences in round $r+2$ will affect strictly more than two T-boxes in round $r+4$. □

Based on these properties, we have a tool to identify restricted bit flips, and to distinguish non-linear T-boxes. In Algorithm 3, this procedure is described. Note that during the algorithm, we also store the differences Δm representing flips of middle bits $(b_4 b_5)$ to an S-box S_m in the set $\mathcal{S}_M(T_m^{r+2})$.

Algorithm 3: Split R_{r-1} into $\overline{R_{r-1}}$ and $R_{r-1}\backslash\overline{R_{r-1}}$ flips

forall $\Delta v \in \mathcal{S}_R(T_j^r)$ **do**
  Compute 2 round functions $\Delta m, \Delta n \leftarrow$ propagated differences in round $r+2$ of T_m^{r+2}, T_n^{r+2}, with $m \neq n$
  $\delta m \leftarrow$ # affected T-boxes in round $r+4$ propagated by Δm in round $r+2$.
  $\delta n \leftarrow$ # affected T-boxes in round $r+4$ propagated by Δn in round $r+2$.
  if $\delta m > 2$ *and* $\delta n = 2$ **then**
    $\Delta v \rightarrow \mathcal{S}_{\overline{R}}(T_j^r); \Delta m \rightarrow \mathcal{S}_M(T_m^{r+2})$
    Denote T_m^{r+2} as non-linear T-box
  end
  else if $\delta m = 2$ *and* $\delta n > 2$ **then**
    $\Delta v \rightarrow \mathcal{S}_{\overline{R}}(T_j^r); \Delta n \rightarrow \mathcal{S}_M(T_n^{r+2})$
    Denote T_n^{r+2} as non-linear T-box
  end
  else if $\delta m > 2$ *and* $\delta n > 2$ **then**
    $\Delta v \rightarrow \mathcal{S}_{R\backslash\overline{R}}(T_j^r)$
    Denote both T_m^{r+2} and T_n^{r+2} as non-linear T-box
  end
end

With the combination of Algorithm 2 and Algorithm 3, applied on T_j^r ($1 \leq j \leq 12$), we are able to obtain the following information of differences on the input to the obfuscated round r:

$$\begin{cases} \mathcal{S}_{\overline{R}}^r = \cup_j \mathcal{S}_{\overline{R}}(T_j^r): & \text{differences representing restricted bit flips} \\ \mathcal{S}_M^{r+2} = \cup_j \mathcal{S}_M(T_j^{r+2}): & \text{differences representing S-box middle bit flips} \\ T_{\pi(1)}^{r+2} \ldots T_{\pi(8)}^{r+2}: & \text{the 8 non-linear T-boxes } (\pi \text{ unknown}) \end{cases}$$

3.8.2 Finding Single Bit Flips

In the previous step, we were able to obtain differences on the input of obfuscated T-boxes, that represent flips of the 2 middle input bits of the underlaying S-boxes. Let T_j^{r+2} be an arbitrary non-linear T-box in round $r+2$, and $\mathcal{S}_M(T_j^{r+2})$ its set of middle bit flips. Denote $\mathcal{S}_M(T_j^{r+2}) = \{\Delta m_1, \Delta m_2, \Delta m_3\}$ the set of obtained differences (two single bit flips, and one double bit flip).

One can verify that, except for S-box S_8, each of the four output bits of the S-box S_j^{r+2} are flipped at least once by going through one of the values $v_j^{r+2} \oplus \Delta m_1, v_j^{r+2} \oplus \Delta m_2, v_j^{r+2} \oplus \Delta m_3$. Furthermore, as the middle bits are not bypassed

in the same T-box, no other output bits of the T-box are affected. Due to the diffusion property of the DES permutation P, each of the four output bits affects a different S-box in round $r+3$ (see [33]). Thus, the propagated input differences to the T-boxes in round $r+3$ represent single bit flips. Algorithm 4 describes this procedure, which constructs the set $\mathcal{S}_S(T_i^{r+3})$ of differences representing single bit flips.

As mentioned, the described property does not hold for S_8: for the input $11b_4b_501$, with arbitrary b_4 and b_5, the rightmost output bit cannot be flipped by flipping the input bits b_4 and b_5. Thus, with a probability of 1/16, we are not able to find all single bit flips of round $r+3$. However, it will become clear in the next section that we do not need all information to successfully apply our cryptanalysis.

Algorithm 4: Finding single bit flips

forall $\Delta v \in \mathcal{S}_M(T_{\pi(j)}^{r+2}) \quad j = 1 \ldots 8$ *(for non-linear T-boxes)* **do**
  Compute one round function
  forall Δw_i *propagated difference to a T-box* T_i^{r+3} **do**
    $\Delta w_i \rightarrow \mathcal{S}_S(T_i^{r+3})$
  end
end

3.8.3 Obtain the Inputs to the S-boxes

Let T_j^{r+3} be an arbitrary non-linear T-box in round $r+3$. Using the acquired information from the steps above, we deploy a filter algorithm to identify the S-box ($S_{\pi^{-1}(j)}$) in the T-box T_j^{r+3}, and to find the value of its 6-bit input vector $(f_j^{r+3}|_{2\ldots7}(v_j^r) \oplus k_j^{r+3})$.

We define the set $\mathcal{P}(T_j^{r+3}) = \{(S_q, w_l) \mid 1 \leq q \leq 8, w_l \in GF(2)^6\}$ as the set of all possible pairs of S-boxes and input vectors. Our strategy is to remove all the invalid pairs from the set. We can do this by comparing the number of affected T-boxes in round $r+4$ when a difference $\Delta v_i \in \mathcal{S}_S(T_j^{r+3}) \cup \mathcal{S}_M(T_j^{r+3})$ is applied to the input of T_j^{r+3}, with the number of affected S-boxes in a non-white-box DES simulation with a pair $(S_q, w_l) \in \mathcal{P}(T_j^{r+3})$.

We define δ_i as the number of T-boxes that are affected when Δv_i is applied. To verify a pair (S_q, w_l), we take part of a non-white-box DES implementation with S-box S_q and S-box input w_l, and simulate the behavior of a flip of the i'th input bit to the S-box. Then, δ_i' is defined as the number of affected S-boxes in the next round of this simulation. Define Δw_i as the difference to the

input of the internal S-box of the T-box to which Δv_i is applied ($\Delta w_i : w_l \rightarrow w_l \oplus f_j^{r+3}|_{2\ldots7}(\Delta v_i)$).

If (S_q, w_l) is a candidate solution, it should satisfy the following conditions:

- There can only be one S_q for each round.
- $\Delta v_7 = \mathcal{S}_M(T_j^{r+3}) \backslash \mathcal{S}_S(T_j^{r+3})$ is the flip of both middle bits, represented as $\Delta w_7 = 001100$, for which δ_7' can be computed. Because Δv_7 only affects bits of Y_{r+3}, δ_7 must be equal to δ_7'.
- $\{\Delta v_3, \Delta v_4\} = \mathcal{S}_M(T_j^{r+3}) \cap \mathcal{S}_S(T_j^{r+3})$ represent the two single flips of the input bits to the S-box, but we do not know in which order. Moreover they only affect bits of Y_{r+3}, and thus we must have $\{\delta_3', \delta_4'\} = \{\delta_3, \delta_4\}$.
- Similarly $\{\delta_2', \delta_5'\} \in \{\delta_1, \delta_2, \delta_5, \delta_6\}$.
- The differences affecting the outer input bits affect bits of R_{r+2}, and therefore the number of affected S-boxes can be smaller than the number of affected T-boxes, which should be taken into account when comparing $\{\delta_1', \delta_6'\}$ to $\{\delta_1, \delta_2, \delta_5, \delta_6\} \backslash \{\delta_2', \delta_5'\}$.

Any pair (S_q, w_l) that does not fulfill these conditions is removed from the set $\mathcal{P}(T_j^{r+3})$. At the end, if only pairs with one type S_q remain, then this S_q is the internal S-box of $T_j^{r+3}(\pi(q) = j)$. As soon as S-boxes are identified, we can also make use of S-box relations between consecutive rounds. E.g., S_1 in round r does not affect S-box S_1 and S_7 in round $r+1$. Moreover, if for example S_3 is identified in round $r+1$, then S_1 affects its second input bit, which allows us to narrow down the conditions ($\delta_2 = \delta_2'$).

Because all eight DES S-boxes are very different, and are highly non-linear, the filtering process will reduce most $\mathcal{P}(T_j^{r+3})$ sets to a singleton (S_q, w_l), where $S_q = S_{\pi^{-1}(j)}$ is the internal S-box and $w_l = f_j^{r+3}|_{2\ldots7}(v_j^{r+3})$ the 6-bit input vector to this S-box.

3.8.4 Key Recovery

Given that we have found a sufficient number of inputs to S-boxes, we start an iterated recovery of key bits, initiated by guessing one single key bit, using the following two observations:

- The expansion operation E maps some of the input bits to 2 different S-boxes, prior to the key addition. In step 3, we have obtained values of the input bits to these S-boxes, after the round key addition. Hence, if we know one of the corresponding two bits of the round key, we are able to compute the other key bit.

- The value of one single bit can be followed through several rounds. Consider an R_{r-1} bit. In round r and $r+2$, after the expansion and the round key addition, this is the (known) input to an S-box. In round $r+1$ it is XOR-ed with an output bit b of an S-box after the permutation P operation. Because P is known, the S-boxes in round $r+1$ are identified and their input is known, we can compute the value of b. Hence, if one bit of the round key bit in round r or $r+2$ is known, we can compute the other key bit.

Iterated use of these algorithms generates the DES key bits. When a new round key bit is computed, we can pull this back through the DES key schedule. This is possible, because the 48-bit round key is a fixed permutation of a subset of the 56-bit DES key. New key bits in turn result into new round key bits, to which the two described methods can be applied.

Depending on the initial key bit guess, two complementary keys k_0 and k_1 can be computed. Because of the complementation property DES exhibits, both keys are a valid result. The complementation property of DES [148] is defined as

$$DES_k = \bigoplus_1 \circ DES_{k\oplus 1} \circ \bigoplus_1 ,$$

where $\bigoplus_1$ represents the XOR with the all one vector. Then

$$\begin{aligned} G \circ DES_k \circ F &= G \circ \bigoplus_1 \circ DES_{k\oplus 1} \circ \bigoplus_1 \circ F \\ &= G' \circ DES_{k\oplus 1} \circ F' . \end{aligned}$$

Hence if k is the original DES key, and F, G the external encodings used to shield the white-box DES implementation, then the complementary key $k \oplus 1$ is also a valid DES key with external encodings F', G', where $F' = \bigoplus_1 \circ F$ and $G' = G \circ \bigoplus_1$.

Our algorithm recovers two possible solutions of keys that could have been embedded in the white-box implementations. Because the external encodings can be chosen freely, there will be no means to discover which of the two secret keys was chosen. Hence we can conclude that the *white-box ambiguity* (see Sect. 3.3.2) for white-box DES implementations is exactly two, i.e., the number of given constructions (secret key + external encodings) which could produce exactly the same set of tables. It will be exactly two, because once the initial key bit has been chosen, the secret key (and the external encodings) can be computed by a deterministic algorithm.

3.8.5 Recovery of the External Encodings

The prior goal in the cryptanalysis of any white-box implementations, is to recover information of the embedded secret key. However, in most cases in practice, to break a system where white-box implementations are deployed, information of the external encodings that have been used will need to be recovered as well. E.g. in DRM systems, white-box implementations will be accompanied with external encodings in order to 'lock' them to authentication code, or a rights management engine. To decrypt protected media content, it will need to be processed by these external encodings as well.

For any white-box DES implementation, as soon as the key has been recovered, the external encodings can be recovered in a generic way as follows: for any given input v_{EXTin} to the encoded implementation, we are able to find the inputs to the S-boxes. We simply deploy the cryptanalysis above, and choose v_{EXTin} as the vector X in the initialization phase.

For Feistel ciphers, given the input to two consecutive rounds and the secret key, the plaintext can easily be computed backwards. Hence we are able to compute the input to the *naked DES*, i.e., $v_{DESin} = F(v_{EXTin})$, where F is the (unknown) input encoding. Once F and the DES key k have been computed, the output of G for any given input v can be computed easily, since $G(v) = \text{WBDES} \circ F^{-1} \circ DES_k^{-1}(v)$. Hence, chosen input-output pairs for the external encodings can be computed. As a result, when the external encodings are learnable, their definition can be recovered. A function is called learnable when a limited amount of input-output pairs satisfy to define the function (see also Definition 11). Examples of learnable functions are linear and affine functions, and polynomials.

Chow *et al.* [43] proposed a specific class of external encodings: block encoded affine mixing bijections. Suppose these block encodings are nibble encodings. To recover the input encoding F, it suffices to run through all 16 possible values for each of the 16 nibble input encoding independently, keeping the other inputs fixed. Once F is recovered, chosen input-output pairs for the external output encoding G can be computed, which eventually leads to the recovery of the definition of the external output encodings recommended by Chow *et al.*.

Implementation and Complexity

We have implemented our cryptanalysis in C++, and conducted tests on a Pentium M 2 GHz. On average, about 2^{13} obfuscated round functions of the white-box DES implementation need to be computed to check the difference propagations. This is less than our complexity study below indicates, due to some

extra optimizations we have applied (e.g., introducing requirements regarding round $r+1$ in Property 1 substantially improves the efficiency of the algorithm). Moreover, our tests indicate that computations with eight consecutive obfuscated round functions is sufficient for the attack to succeed. There is no restriction on which window of eight round functions to chose. The space complexity is negligible, as most space is used in step three of the cryptanalysis, to store the set $\mathcal{P}(T_j^r)$ of candidate pairs (S_i, w_l). Note that we can pre-compute the simulates set of pairs, because that does not require any information of the key and implementation.

In the conducted tests on several white-box DES implementations, our cryptanalysis algorithm extracted the DES key in all tests in under a second. On average the cryptanalysis requires 0.64 seconds on a standard 2Ghz computer.

Complexity. We define the complexity of the cryptanalysis as the number of round functions of the white-box implementation that need to be computed. The first step described, to retrieve flips of bits of R_{r-1}, has the highest complexity. Because of the lack of any prior information on internal flips, all differences have to be computed through several rounds in order to learn this bit flip information.

In Algorithm 2, for all twelve T-boxes, and all $2^8 - 1$ possible differences, two rounds need to be computed to observe the difference propagation. This corresponds to a total of $12 \cdot (2^8 - 1) \cdot 2 = 6120$ round function computations. For each positive result, we perform at most two double checks as described in Property 2. Algorithm 3 requires six round computations for each difference of $\mathcal{S}_R$ (two for Δv, two for Δl and two for Δm). Hence, 240 round functions computations are performed.

Consequently, we can retrieve all flips of bits of $\overline{R_{r-1}}$ for one round in less than 2^{13} round computations in total. As described in Property 2, from $\Delta v \in \mathcal{S}_{\overline{R}}^r$, we can efficiently compute $\Delta n \in \mathcal{S}_{\overline{R}}^{r+2}$. Because of the one-to-one relation between Δv and Δn, this is sufficient to find all the single $\mathcal{S}_{\overline{R}}^{r+2}$ bit flips. Thus, when for two consecutive rounds, $\mathcal{S}_{\overline{R}}$ is found, we can compute this set for all subsequent rounds using Property 3 only. Hence, with about 2^{14} round computations, we can compute all flips of single bits of $\overline{R_{r-1}}$ for all rounds.

The complexity of the other steps of the cryptanalysis is negligible. In Algorithm 4, for each $\Delta m \in \mathcal{S}_M^r$, one round function needs to be computed. Hence, for each round, at most 24 round computations are needed (for 16 single bit flips and at most 8 double bit flips). To compute the exact inputs to the S-boxes, a filtering process needs to be applied to each non-linear T-box. In the worst case, we need to compute the difference propagation for all seven input differences. Thus at most seven round computations for each of the eight non-linear T-boxes. The simulation process for each T-box needs to be performed at most $2^6 \cdot 8 = 2^9 (= |\mathcal{P}(T_j^r)|)$ times, which is the equivalent effort of computing one

white-box DES round function (which consists of $552 \sim 2^9$ lookup table computations). The total complexity to compute the inputs to all S-boxes of one round is thus $8 \cdot (7 + 1) = 2^6$.

Hence, the total complexity to recover a representation of the secret key embedded in white-box DES implementations is 2^{14}.

3.9 Conclusion

White-box implementations are hard to understand, and their techniques are deployed in different ways on different ciphers. Moreover, their security is hard to evaluate, while only indicative metrics exist. As a consequence there is a strong need for cryptanalytic work to investigate the robustness of the techniques used in white-box implementations.

The cryptanalysis on white-box DES implementations can be mounted for a great deal due to properties that are specific to the DES. The confusion property of the DES S-boxes, the diffusion property of the DES permutation P, and the design of the expansion operation are critical operations used to extract key information. It appears to be difficult to obfuscate their behavior, and it has been shown that a sufficient amount of information leaks to recover the secret key.

The analysis, while specific to the DES, nevertheless points the way to analyze other ciphers. Feistel ciphers are particularly vulnerable due to their inherent unbalanced differential propagation , which was abused in recent cryptanalysis [191]. Within this context, a practical measure to estimate the vulnerability of a block cipher to differential cryptanalysis on its white-box implementation, would consist in computing the minimum number of active S-boxes . A low bound on this metric might indicate a point of attack. Studies on the black-box security of block ciphers against differential cryptanalysis have conducted similar research (on active S-boxes). As a result, families of cryptographic ciphers with better differential properties have been proposed, such as the family of ciphers with diffusion based on MDS matrices. It would seem to be that these ciphers are better candidates to obtain secure white-box implementations.

Cryptanalysis of white-box AES implementations have presented algebraic attacks. In particular, it has been shown by Michiels *et al.* [135] that MDS matrices introduce an attack point to mount algebraic analysis.

In conclusion to these cryptanalytic results on white-box implementations, we observe a conflict between black-box security criteria and white-box security.

Black-box security criteria are introduced based on black-box analysis of cryptographic primitives. A first criterion is high non-linearity, to protect against linear attacks. This is fulfilled by the use of S-boxes in common block ciphers.[7] Other criteria that are introduced are balancedness, algebraic degree, optimal diffusion (for which MDS matrices are introduced), active S-boxes, and correlation immunity.

White-box security criteria at this point are mainly obtained by cryptanalytic experiences, and white-box metrics such as diversity and ambiguity. The latter metric is the most important from a security point of view, since it accounts for the space of possibilities which an adversary must disambiguate in order to find key information. Unfortunately, metrics can only be indicative, hence the importance of cryptanalytic techniques. The cryptanalysis results have shown that some cryptographic building blocks introduce a point of attack. For example, MDS matrices have been exploited in the generic attack by Michiels *et al.* [135]. Hence, a building block that has been introduced because of its properties against black-box cryptanalysis, is exactly the building block that enables an attack on its white-box implementation. The interesting conclusion is that, although white-box cryptography investigates techniques on how to *compile* a block cipher in a secure way, it also imposes requirements on the design of the block ciphers.

We have also extended the analysis to basic building blocks. In particular, addition tables are prone to analysis. We have shown to extract information on the encodings of an addition table, and how this can be used as an alternative attack on the white-box implementations of AES and DES.

3.9.1 Further Research

The current white-box implementations of the DES and the AES have shown to be insecure. To thwart their cryptanalysis, slight modifications in the white-boxing techniques might be introduced, certainly for the DES. However, our impression is that no matter which improvements on white-box implementation techniques are presented, the white-box implementations of some ciphers will remain insecure. This because of the observation that some block cipher building blocks might yield a point of attack.

Instead, white-box implementations of ciphers composed of other building blocks must be studied; even new block ciphers might be proposed that are suitable for white-boxing, while providing a sufficient level of black-box security. Therefore, we should move forward towards a public scrutiny process for white-box implementations. New ideas and constructions on the design of white-box

[7] The non-linearity of an S-box is the minimal distance of all non-trivial combinations of the columns of S to the set of affine functions. We refer the work by Braeken [30] for an extensive study on the construction of S-boxes.

implementations need to be presented and published at cryptology conferences, although they are not accompanied by security proofs. This way, one can learn what exactly makes white-box implementations (in)secure, and learn how to improve such implementations. One should of course keep black-box security criteria in mind, because white-box implementations need to be secure against black-box attacks in the first place. Unfortunately, white-box implementations are not yet well accepted by the cryptology community, and theoretical results are often miss-interpreted [6].

Based on our discussions above, we offer the following reflections on requirements and methodologies for new white-box implementations:

- White-box implementations should not only prevent key recovery, but also be non-invertible. Hence, white-box implementations introduce some natural asymmetric behavior (even if they are applied to symmetric ciphers). One can encrypt plaintexts, given a white-boxed encryption implementation, while only the party that has knowledge of the secret key, or has a decryption implementation is able to do so. Therefore, building blocks for new white-box implementations can potentially be found in the field of asymmetric cryptography. A block cipher based on asymmetric primitives has been proposed by Billet *et al.* [20] and is an interesting direction for further research.

  Note that it is not an absolute requirement to implement white-box implementations as a network of lookup tables. Constructions based on polynomials can be proposed, where key information is hidden by introducing obfuscating polynomials, similar to the research in *Hidden Field Equations* (HFE), *Isomorphisms of Polynomials* (IP) [145], and *Multivariate Equations* (MQ) [188].

- For practical purposes, the implementation size of white-box implementations matters. In one publication, Bringer *et al.* [32] have presented a white-box AES implementation of around 568 MB. Certainly when white-box implementations are deployed on embedded devices (to prevent side-channel analysis), these constraints have to be taken into consideration.

- In a white-box attack context, many more algebraic equations can be constructed than in a black-box attack context. To thwart algebraic analysis, the algebraic (round) structure should be destroyed. Bringer *et al.* [32] have introduced a perturbations approach on white-box AES implementations. Another approach is to construct round functions based on non-compatible operators; a similar approach has been conceived in the design of the IDEA block cipher [115]. Several non-compatible operators can be combined into

a (key dependent) lookup table, making it hard to analyze. These lookup tables should have a small dimension input, while their output size can be arbitrary.

Key update

Once secure white-box implementations have been constructed, an interesting direction for further study, is how to manage key updates. That is, given a white-box implementation of E_k, how can a minimal amount of modifications yield an implementation of $E_{k'}$, while preserving the security.

Consider for example the white-box AES implementation as described in Sect. 3.5.2. A key update could be achieved by injecting a new type of tables before the input of Type II tables (see Fig. 3.5). Let n_1, n_2 be the two nibble input encodings to this Type II table, and denote this new type of tables as Type V, which can be defined as

$$\text{Type V}_{\Delta k_j^r} = (n_1^{-1} \| n_2^{-1}) \circ \{8 \times 8\}^{-1} \circ \bigoplus_{\Delta k_j^r} \circ \{8 \times 8\} \circ (n_1 \| n_2)\,,$$

where Δk_j^r defines the 8-bit round sub-key modification. Then, 80 kbytes would be sufficient to update an AES key. Unfortunately, this approach is quite limited, since the information of multiple key updates might offer sufficient information to recover information on the nibble encodings, similar to the attack described by Billet *et al.* [21]. Let Q denote $(n_1 \| n_2) \circ \{8 \times 8\}$, then

$$\left[\text{Type V}_{\beta_1}\right] \circ \left[\text{Type V}_{\beta_2}\right]^{-1} = Q \circ \bigoplus_{\beta_1 \oplus \beta_2} \circ\, Q^{-1}\,.$$

Hence, eight linear independant round key updates gives the entire space of possible $\beta \in \text{GF}(2^8)$ (think of more combinations such as $\left[\text{Type V}_{\beta_1}\right] \circ \left[\text{Type V}_{\beta_2}\right] \circ \left[\text{Type V}_{\beta_3}\right]^{-1}$), such that Theorem 1 can be used to recover information on the encoding Q.

Therefore, more complex updates will be required. But first, secure white-box implementations need to be presented.

Chapter 4

A Theoretical Model for White-Box Cryptography

4.1 Introduction

White-Box Cryptography deals with protecting cryptographic primitives embedded in a program to which an attacker has white-box access. It aims to provide security when the program is executing in a hostile environment and the attacker can conduct non-black-box attacks (such as code inspection, execution environment modification, code modification, etc). Practical white-box implementations of DES and AES encryption algorithms were proposed by Chow *et al.* [43, 42], which we have studied in Chapter 3. However, no formal definitions of white-box cryptography were given, neither were there any proofs of security. With their subsequent cryptanalysis [21, 191, 82], it remains an open question whether or not such white-box implementations exist.

One way to realize WBC is to obfuscate (using an obfuscator) the executable code of the algorithm and hope that the adversary cannot use it in a non-black-box manner. What we would like is that, given an obfuscator satisfying some definition, a white-box implementation can be proved secure under some *security notion*. While code obfuscation attempts to hide certain characteristics of a program P, white-box cryptography specifically focusses on software implementations of cryptographic primitives (such as encryption schemes); its goal is to offer a certain level of robustness against an adversary who has full access to and control over the implementation of the primitive. Several models for obfuscation have been presented before, but it is not clear if any of these definitions can capture the concept of white-box cryptography. Other models that have been presented include the malicious host model [94, 162, 163], within the context of

mobile agents.

In this chapter, we discuss the shortcomings of obfuscation models, their relation with white-box cryptography, and formalize the notion of white-box cryptography by capturing the security requirement using a 'White-Box Property' (WBP). We elaborate on the work we have presented in [165]. This new theoretical model provides a context to investigate the security of white-box implementations. The chapter is organized as follows: in Sect. 4.2, we introduce the concept of code obfuscation and discuss the most important (im)possibility results. The formal definitions of code obfuscation are provided in Sect. 4.6. Since these models fail to capture the concept of white-box cryptography, we present a new model in Sect. 4.7. A key concept in our model are *security notions* (Sect. 4.3), which have been presented as a concept to capture the requirements of cryptographic primitives in the 'black-box' model, and are introduced for the formalization of theoretical security proofs [72, 75]. Inspired by theoretical results on obfuscation, we translate some theoretical properties onto our model, and present (im)possibility results in Sect. 4.8.

4.1.1 Related Work

The notion of code-obfuscation was first given by Hada in [84], who introduced the concept of *virtual black-box property* (VBBP) using computational indistinguishability. In [6], Barak *et al.* defined obfuscation using the weaker predicate-based VBBP and showed that there exist unobfuscatable function families under this definition. Goldwasser and Kalai [76] extend the impossibility results of [6] w.r.t. auxiliary inputs.

On the other hand, several positive results have been formalized. For instance, Lynn *et al.* show in [122] how to obfuscate point functions in the random oracle model [10]. Subsequently, Wee [186] demonstrated how to obfuscate point functions without random oracles. Hohenberger *et al.* [93] used a stronger notion of obfuscation (average-case secure obfuscation) and showed how it can be used to prove the security of re-encryption functionality in a weak security model (i.e., IND-CPA). They also presented a re-encryption scheme under bilinear complexity assumptions. Hofheinz *et al.* [91] discuss a related notion of obfuscation and show that IND-CPA encryption and point functions can be securely obfuscated in their definition. Goldwasser and Rothblum [81] define the notion of "best-possible obfuscation" in order to give a qualitative measure of information leakage by an obfuscation (however, they do not differentiate between "useful" and "useless" information). Recently, Canetti and Dakdouk [35] gave an obfuscator for point functions with multi-bit output for use in primitives called "digital lockers". Finally, Herzberg *et al.* [89] introduce the concept of *White-Box Remote*

Program Execution (WBRPE) in order to give a meaningful notion of "software hardening" for all programs and avoid the negative results of [6].

4.1.2 Terminology

Formal models in computer science are often described on *Turing Machines*, which are abstract devices which can simulate any computer algorithm [181]. They are only intended to provide a basic logic for formal theories in computer science, and hence are not constructed in practice. Denote by $\mathbb{TM}$ the set of all Turing Machines (TMs). Studying their abstract properties yields many new insights in computer science, which is also our objective within the context of white-box cryptography. In electrical engineering, often a formal model based on circuits is used. Note that for simplicity, in this work, we only consider the case of Turing Machines. However, all results also carry over to circuits.

For any TM X, we denote by $|X|$ the length of the string containing a description of X. For simplicity, we define the input-space of arbitrary TMs to be $\{0,1\}^*$, the set of all strings. If however, the input-space of a TM is well defined and efficiently samplable (for instance, the strings should be of a particular encoding), then we implicitly imply that the inputs are chosen from the input-space sampled using a string from $\{0,1\}^*$. All our definitions and results apply in this extended setting without any loss of generality.

Denote by $\mathbb{P}$ the set of all polynomials with non-negative integer coefficients. A mapping $f : x \in \mathbb{N} \mapsto f(x) \in \mathbb{R}$ is a *negligible function* in x (written $f(x) \leq neg(x)$) if

$$\forall p \in \mathbb{P}, \exists x' \in \mathbb{N}, \text{ such that } \forall x > x' : f(x) < 1/p(x)\,.$$

A *point function* f_α is a function that evaluates to 1 if and only if α is presented as input, 0 for any other input.

$$f_\alpha(x) = \begin{cases} 1 & \text{iff } x = \alpha \\ 0 & \text{otherwise}\,. \end{cases}$$

Denote with $A^P(x)$ some primitive A that computes on input x and is given oracle access to P.

These definitions are used in the sections below. In Sect. 4.5, more definitions are provided, which are needed to formalize the model for white-box cryptography that we present.

4.2 Code Obfuscation

Code obfuscation is the most viable method to prevent reverse-engineering [45]. A code obfuscator is used to convert a code (program) into an equivalent one that

is difficult to reverse engineer, by distinguishing its internal workings. We denote an obfuscator as $\mathcal{O}$, and the obfuscation of the program P as $\mathcal{O}(P)$. In this section, we review the theoretical approaches on code obfuscation, upon which our model for white-box cryptographic is inspired. For an overview of (practical) code obfuscation techniques, we refer to our survey in [38].

4.2.1 Definitions for Obfuscation

The first contributions towards a formalization of code obfuscation were made by Hada [84], who presented definitions for obfuscation based on the *simulation paradigm* for *zero knowledge*, called GMR-ZK, given in [74].

Using simulation is an approach that has been used before in formal security proofs for cryptography. In this approach, there are two settings. One setting in which an arbitrary, probabilistic, polynomial-time adversary can interact with the cryptographic primitive; in the other setting, the adversary interacts with an idealized version of the cryptographic primitive that can never be broken. Note that this idealized version is an abstract primitive. E.g., a perfectly secure encryption function as an idealized abstract version of a practical encryption scheme. To determine whether a primitive is secure, the output of the adversaries in the two settings is compared. If their outputs are approximately the same (or indistinguishable in the case of output distributions), then the cryptographic primitive must be secure, since the idealized version is secure.

The main difference between the obfuscation definition and the simulation based definitions used in (black-box) cryptography, is in the type of objects the adversary interacts with. In the obfuscation case, it is a comparison between (white-box) interaction to an implementation of the primitive, and the interaction with a oracle implementation (black-box) [5]. In the tradition cryptography case, it is between an oracle implementation of the cryptographic primitive, and an idealized version. This new concept is captured by the *Virtual Black-Box Property (VBBP)*.

This brings us to the definition of obfuscation, which was formalized by Canetti [34] (for point functions) and Barak *et al.* [6] (for any function).

Definition 5 (Obfuscator) *A* probabilistic algorithm $\mathcal{O}$ *is an obfuscator if the following three conditions hold:*

- **Functionality** – $\forall$ *program* P, $\mathcal{O}(P)$ *describes a program that computes the same function as* P*.*
- **Polynomial slowdown** – *There exists a polynomial* p*, such that* $\forall P : |\mathcal{O}(P)| \leq p(|P|)$*, and if* P *halts in* t *steps on some input* x*, then* $\mathcal{O}(P)$ *halts in* $p(t)$ *steps on input* x*.*

- **Virtual Black Box Property** – *Given access to the obfuscated program $\mathcal{O}(P)$, an adversary should not be able to* learn anything *about the program P, that it could not learn from oracle access to P.*

Informally, an *obfuscator* $\mathcal{O}$ is an (efficient, probabilistic) "compiler" that takes as input a program (or circuit) P, and produces a new program (respectively circuit) $\mathcal{O}(P)$ that has the same functionality as P yet is "unintelligible" in some sense. This property of "unintelligible" is captured in Definition 5 by the Virtual Black-Box Property, and has been defined in several different formulations by Barak *et al.* [6]. We present the main notions and their comparison below.

Predicate-based notion of obfuscation

In this notion, an adversary aims to compute *some predicate* on the program P. In this sense, the virtual black-box property captures that for any adversary and any Boolean predicate π, the probability that an adversary is able to compute $\pi(P)$ given the obfuscation $\mathcal{O}(P)$ should be comparable to the probability that a simulator S is able to compute $\pi(P)$ when given only oracle access to P. Roughly speaking, this guarantees that the adversary A does not have any advantage of white-box access, compared to a black-box simulation, hence the obfuscation does not leak any extra information on $\pi(P)$. Definition 6 formally captures this notion.

Definition 6 (Predicate-based Virtual Black-Box Property) *An obfuscator $\mathcal{O}$ satisfies the Predicate-based VBBP if for any predicate π and for any (polynomial time) adversary A, there exists a (polynomial time) simulator S, such that for all programs $P \in \mathcal{P}$:*

$$\left| \Pr\left[A(1^s, \mathcal{O}(P)) = \pi(P)\right] - \Pr\left[S_A^P(1^s) = \pi(P)\right] \right| \leq neg(|P|)\,,$$

where the probabilities are taken over the coin tosses of A, S, and $\mathcal{O}$, and s is some security parameter (often defined in terms of $|P|$).

Note that the predicate does not need to be efficiently computable, and that the notion needs to be qualified over *all* adversaries. This notion of obfuscation is depicted in Figure 4.1, where (a) depicts the white-box adversary A with access to $\mathcal{O}(P)$, and (b) the simulator which has oracle access to P, and has A as a subroutine (to be able to simulate its behavior).

However, as pointed out by Barak *et al.* [6] and Hohenberger *et al.* [93], we might consider a stronger notion of "virtual black-box". The predicate definition

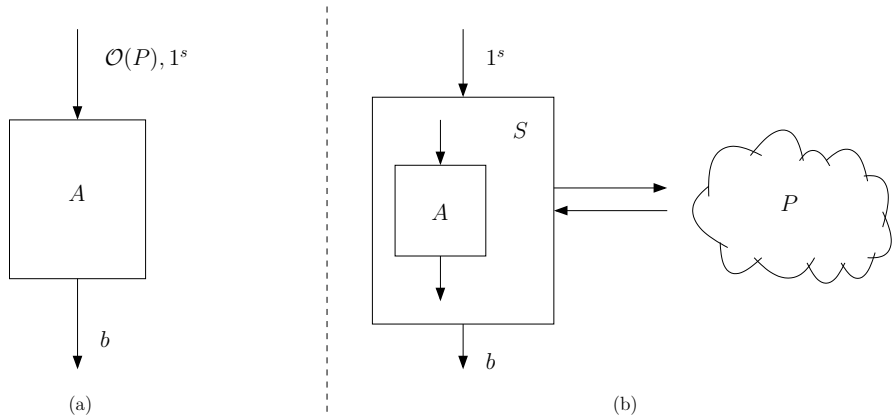

Figure 4.1. Predicate-based Virtual Black Box Property

does give some quantifiable notion that some information (i.e., predicates) remains hidden, but other non-black-box information might leak and compromise the security of the system.

Distinguisher-based notion of obfuscation

This notion of obfuscation is based on *computational indistinguishability*, and does not restrict what the adversary is trying to compute. We say that, for any adversary given the obfuscated program $\mathcal{O}(P)$, it should be possible to construct a simulator S (with only oracle access to P) that is able to produce a similar output. This notion of similarity is captured by a distinguisher D.

Definition 7 (Distinguisher-based Virtual Black-Box Property) *An obfuscator $\mathcal{O}$ satisfies the distinguisher-based VBBP if for any (polynomial time) adversary A, there exists a (polynomial time) simulator S, such that for all programs $P \in \mathcal{P}$:*

$$\left| \Pr\left[D(\mathcal{O}(P)) = 1\right] - \Pr\left[D(S^P(1^{|P|})) = 1\right] \right| \leq neg(|P|)\,,$$

where D is a distinguisher, and the probabilities are taken over the coin tosses of A, S, and $\mathcal{O}$.

This notion of security is quite similar to the notion of semantic security for (black-box) cryptographic schemes. As pointed out by Wee [186], this removes the need to quantify over all adversaries, as it is necessary and sufficient to simulate the output of the obfuscator. To avoid trivial obfuscation, Hofheinz *et al.* [91]

extended the distinguisher-based definition by giving the distinguisher oracle access to the functionality P. A similar approach was conceived by Hada [84], with the difference that an extra copy of the obfuscation was presented to the distinguisher, instead of oracle access to the functionality. This leads to a very strong notion of obfuscation. We will discuss this in more detail below.

Other notions of obfuscation

Barak *et al.* [6] presented other notions of obfuscation, which are largely defined over what the adversary aims to compute. The following are the most common ones (in decreasing order of generality)

- **Computational indistinguishability** [6, 91, 93, 186] *(strongest security requirement)* – Given just oracle access to P, an adversary tries to produce an output distribution that is computationally indistinguishable from what the adversary would compute when given $\mathcal{O}(P)$.
- **Satisfying a relation** [6] – Adversary tries to produce an output that satisfies an arbitrary relation with the original program.
- **Computing a function** [6] – Restrict the previous requirement to relations which are functions.
- **Computing a predicate** [6, 186] *(weakest security requirement)* – restrict previous to {0,1}-valued functions; that is, the adversary trying to decide if some property of the original program is true or false.

Different notions of obfuscation lead to different results. Below, we present some (im)possibility results. Note that impossibility results have the strongest meaning under the weakest notion of obfuscation, while positive results have the strongest meaning under the strongest notion of obfuscation. Also, these definitions of obfuscation can be extended towards *approximate obfuscators*.

Definition 8 (Approximate obfuscator) *A predicate/distinguisher-based approximate obfuscator $\mathcal{O}$, is a predicate/distinguisher-based obfuscator that satisfies the following functionality property.*

With an overwhelming probability, $\mathcal{O}(P)$ behaves exactly identical to P on almost all inputs. That is, there exists a negligible function, such that for any input length n:

$$\forall x \in \{0,1\}^n : \Pr\left[\mathcal{O}(P)(x) \neq P(x)\right] \leq neg(|P|)\,,$$

where the probabilities are taken over the coin tosses of $\mathcal{O}$.

4.2.2 Impossibility Results

In their seminal paper [6], Barak *et al.* show that it is impossible to achieve the notion of obfuscation according to Definition 6, that is, it is impossible to construct a *generic obfuscator* for all family of programs (TMs and circuits) $\mathcal{P}$. This is proved by constructing a family of functions $\mathcal{F}$ which is *inherently unobfuscatable* in the sense that there exists some predicate $\pi : \mathcal{F} \to \{0,1\}$ that can be computed efficiently when having access to an obfuscated implementation $\mathcal{O}(f)$ of $f \in \mathcal{F}$, but no efficient simulator can compute $\pi(f)$ much better than by random guessing, given solely oracle access to f.

This results follows from the following paradox.

- If one-way functions exist, then there exists an inherently unobfuscatable function ensemble (under Definition 6).
- The existence of an *efficient* obfuscator implies the existence of one-way functions.

As a result of the above, it can be concluded that efficient obfuscators do not exist.

Because this impossibility applies to the weakest notion of obfuscation, it directly applies to any other (stronger) notion of obfuscation. The presented construction requires that the obfuscated program $\mathcal{O}(P)$ computes exactly the same function as the original program P. With some additional steps, the impossibility results also extends to approximate obfuscators.

An inherently unobfuscatable function family ensemble

The essence of the proof is that there is a fundamental difference between getting oracle access to a function, and having a program that computes it, no matter how it is obfuscated. In order to exploit this, an unobfuscatable function f must not be *learnable*. Obviously, if a function is learnable (i.e., one can acquire a function f' with the same functionality as f by sampling a limited number of oracle queries to f; see also Definition 11), the difference disappears. This implies that learnable functions are trivially obfuscatable according to Definition 5.

Based on the existence of one-way functions, a *specific* family of functions can be constructed. Namely, a family of *cannibalistic functions*, which leak their secret solely when receiving their very own description (functionality) as input. The intuition is that with black-box access a (exact) description cannot be retrieved (because the function is not learnable), while an adversary with access to the obfuscated function (which has by definition the same functionality), can clone the

code and present this as input.[1] Algorithm 8 gives an intuition of such a family of cannibalistic functions. This strategy applies to Turing Machine obfuscators. For circuit obfuscators, a inherently unobfuscatable circuit can be constructed based on a homomorphic encryption oracle. We refer to [6] for technical details on both constructions.

Unobfuscatable cryptographic primitives

Other unobfuscatable families of functions that are presented by Barak *et al.* include cryptographic primitives. Namely, every cryptographic primitive that implies the existence of a one-way function, implies the existence of a respectively unobfuscatable primitive (which can be constructed by using the obtained one-way function as a building block for the primitive) [96]. This applies to digital signature schemes, symmetric-key encryption schemes, pseudo-random function ensembles, and MAC algorithms. In [84], Hada already showed the impossibility of securely obfuscating pseudo-random functions (in the context of their definition of obfuscation).

Remark. Note that the impossibility results refer to *generic obfuscators*. This means that one single (efficient) obfuscator for an entire class of programs (all programs, or all signature schemes) cannot exist. This does not rule out that obfuscators for *specific* (useful, natural) classes of programs cannot exist (there *are* obfuscatable families [6, 186, 91, 165]). Unfortunately, the results by Barak *et al.* are often interpreted incorrectly, and the title (*"On the (Im)possibility of Obfuscating Programs"*) might be misleading. Instead, it should be interpreted as *"On the Impossibility of a Generic Program Obfuscator"*.

Possible directions of further research in response to this impossibility results by Barak *et al.* are (a) to explore weaker but nonetheless meaningful notions of obfuscation, or (b) constructing obfuscators for restricted, yet still non-trivial and interesting classes of programs.

Relaxed notion of obfuscation

Goldwasser and Kalai [76] augmented the predicate-based notion of obfuscation to hold in the presence of auxiliary input. They observe that this is an important requirement for many applications of obfuscation, because auxiliary input comes into play in the real world. Two different definitions are formulated. One definition models security in the presence of auxiliary information that *depends* on the particular function to be obfuscated (for example, previous versions of the software), and one that relates to *independent* auxiliary information.

[1] Note that this cloning trick is very specific to the white-box attack context, and is exactly what is indirectly abused in many impossibility results (e.g., in Theorem 2).

The notion of obfuscation is relaxed in the sense that it only requires security for a random function chosen from some distribution. This is justified by the observation that in most cryptographic applications, an adversary is confronted with such a randomly chosen (obfuscated) function. Whereas requiring the black-box property to hold when an auxiliary input is given, is a strengthening of the original definition by Barak *et al.*, the fact that we require the black-box property to hold for a *random* circuit is a weakening of the requirements of Barak's definition. Weakening the definition of obfuscation, strengthens any impossibility result.

This weakening is meaningful in cryptographic applications, where a class programs $\mathcal{P}$ corresponds to a class of cryptographic algorithms and a random choice of $P \in \mathcal{P}$ corresponds to the choice of a particular secret key to instantiate a cryptographic algorithm.

Goldwasser and Kalai show that relaxing the notion of obfuscation of Barak, does not help to prevent the impossibility result. They prove that there exist many natural classes of functions that cannot be obfuscated with respect to auxiliary input (both dependent and independent auxiliary input). This is shown for *filtered functions* (functions that are forced to output 0 when the input is not of special form). On the positive side, they prove that any obfuscator for the class of point functions is also an obfuscator with respect to independent auxiliary input.

Obfuscating deterministic programs

Wee [186] explored obfuscation of *deterministic programs* under the strong (distinguisher-based) notion of obfuscation, and concluded that deterministic functions can be obfuscated if and only if the function is *learnable* (it is not so hard to see this). A similar result is conceived in [84]. This is partly because these definitions require security for all keys in a given set. Hence, this notion of obfuscation is clearly too strong to achieve any meaningful results. Note that impossibility results in the strong notion of obfuscation are weak, and do not imply impossibility in weaker models of obfuscation.

Following these results, Hofheinz *et al.* [91] remarked that any family of deterministic functions must be approximately learnable to be obfuscatable (in their augmented strong notion of obfuscation). Hence, it is not possible to obfuscate (deterministic) pseudo-random functions under their definition. This in particular rules out the obfuscation of deterministic digital signatures schemes, public-key decryption and pseudo-random functions. Point functions remain possible, as these are *approximate learnable*. Hence the interest to investigate the obfuscatability of *probabilistic* functions, as conceived in [93]. In Sect. 4.9, we introduce a formal definition of probabilistic functions within the context of white-box cryptography.

4.2.3 Positive Results

The results by Barak *et al.* [6] and Goldwasser and Kalai [76] are rather complex. It remains an open question whether obfuscation can be achieved for some natural class of functionalities and circuits. The *halting problem* and *satisfiability* seems to indicate that source code access yields few significant capabilities, apart from black-box access.

A positive result on obfuscation was presented prior to the first formulation of definitions for obfuscation. In [34], Canetti presented a special class of functions suitable for obfuscation under very strong computational assumptions, that works for (almost) arbitrary function distributions. In subsequent work, Canetti *et al.* [37] presented a construction suitable for obfuscation under standard computational assumptions, that is proved secure for uniform function distribution. Both results are probabilistic and technically very sophisticated.

Obfuscation in the random oracle model

Lynn *et al.* [122] explored the question of obfuscation within the *random oracle model* [10]. This is an idealized setting, in which all parties (including the adversary) can make queries to a *random oracle* (see also Sect. 5.1.2). Their results apply to the predicate-based notion on obfuscation. The obfuscatable family of functions that is presented, are point functions. Random oracles can obfuscate point functions because the output of a random oracle $\mathcal{R}$ hides all information about the input that produced it. The obfuscation of the point function f_α is defined as follows:

$$\mathcal{O}(f_\alpha)(x) = \begin{cases} 1 & \text{iff } \mathcal{R}(x) = \alpha' \\ 0 & \text{otherwise}\,, \end{cases}$$

where $\alpha' = \mathcal{R}(\alpha)$, and f_α denotes the point function that evaluates to 1 if and only if α is presented as input. This is closely related to a password verification system.

The idea of using random oracles for obfuscation was originally motivated by the hope that, given access to such an idealized building block, it would be feasible to obfuscate some functionalities. However, Goldwasser and Rothblum [81] presented a strong negative result for obfuscation in the random oracle model. They showed that the existence of such idealized boxes, allows the construction of a natural class of functions that are impossible to obfuscate. Note that the impossibility results presented in [6, 76] extend to the random oracle model.

Obfuscation in the standard model

However, in practice (programmable) random oracles are difficult to realize. Under cryptographic assumptions, it is known how to obfuscate point functions

without a random oracle (in the *standard model*). Canetti [34] showed how to obfuscate point functions (even under a strong auxiliary input definition), using a strong variant of the Decisional Diffie-Hellman assumption. Wee [186] presented a point function obfuscator based on the existence of one-way permutations that are hard to invert in a very strong sense. Their results apply to a weakened predicate-based notion of obfuscation, a relaxation on Definition 6 in allowing the size of S to depend on ϵ, where $\epsilon(n) = neg(n)$. Wee also presented a construction for obfuscating point functions with multi-bit output. These are point functions $f_{\alpha,\beta}$ that evaluate to β on input α, and to 0 on any other input.

However, Canetti and Dakdouk [35] uncovered some weaknesses in Wee's definition. One weakness is that the definition does not provide any security guarantee even under very weak forms of composition. Canetti and Dakdouk augmented this concept and introduced the concept of *digital lockers* and constructed a connection with symmetric encryption as follows: to encrypt a message m using a key k, simply output an obfuscation $F_{k,m}$. We elaborated on this work to construct an asymmetric primitive and improve the bounds related to a conditional adversary; i.e., an adversary limited in terms of number of queries to a random oracle [65]. Similar relations and bounds were discovered by Yao and Yin [197], within the context of *key derivation functions* [114]. A generalization of one-way functions can be found in the work of Dodis and Smith [61] who show how to obfuscate a proximity function.

Improved models

Most of the obfuscation definitions presented above, are either too weak for or incompatible with cryptographic applications, have been shown impossible to achieve, or both. Hohenberger *et al.* [93] and Hofheinz *et al.* [91] present new definitions which have a potential for interesting positive results. On the one hand, the new definitions need to allow the obfuscation of point functions (non-learnable functions), yet at the same time should be strong enough to achieve meaningful results (e.g., turn a symmetric encryption scheme into an asymmetric encryption scheme).

To address this issue, Hohenberger *et al.* introduce the notion of *average-case secure obfuscation*, based on a distinguisher-based definition that allows families of circuits to be probabilistic. Their goal is to capture a similar definition as our Definition 3, and as such, they note that the predicate-based definition cannot guarantee *security* when obfuscated programs are used in cryptographic settings. Hohenberger *et al.* present a *probabilistic* re-encryption functionality that can be securely obfuscated according to this new definition.

Similarly, Hofheinz *et al.* present another variant of a distinguisher-based definition. The deviation is that they consider probabilistic functions and select the

function to be obfuscated according to a distribution. Their new notion is coined *average obfuscation*. The goal is to consider obfuscations for *specific* applications, and they demonstrated the obfuscation of an IND-CPA secure symmetric encryption scheme that results into a IND-CPA secure asymmetric scheme. Similar results hold for the obfuscation of MAC algorithms into digital signature schemes. Note that this result does not apply to IND-CCA secure schemes!

Obviously, the impossibility results by Barak *et al.* [6] apply to these weakened (but still strong) notions.

Other classes

Other work on obfuscation includes Ostrovsky and Skeith [143], who consider a notion of public-key obfuscation applied to keyword search. Such an obfuscator does not maintain the functionality of a program, but rather ensures that the outputs of a public-key obfuscated program are encryptions of the original program's output. A similar variation was considered by Adida and Wikström [2] with their definition of public mixing. These deviate from the original notion of obfuscation in which functionality needs to be retained. A similar strategy is conceived in Herzberg *et al.* [89], in their construction for a remote program execution model, where the output of a virtual machine is an encrypted authenticated output corresponding to some program and input that are both delivered (in encrypted form) to a virtual machine (see also Sect. 5.4.3).

4.2.4 Conclusion

Several definitions of obfuscation have been proposed. They differ in their formulation of the "virtual black-box property" (predicate-based vs. distinguisher-based), and the functionality description (exact vs. approximate functionality). Also, *average case* definitions have been proposed, which capture that the definitions should hold for a *randomly* selected function of the family (according to some distribution), inspired by a random key selection to instantiate cryptographic primitives.

A brief overview of the main results on theoretical obfuscation is presented in Table 4.1. Note that only the main results of a selection of papers are presented.

Although many definitions of obfuscation have been proposed in the literature, none of them is accepted as the default one. One of the problems is as follows: the distinguisher-based definition is too strong in practice to yield any results [93, 186], since deterministic functions can only satisfy the obfuscation definition when they are learnable. This rules out an obfuscator satisfying the distinguisher-based definition for most interesting (deterministic) function families such as pseudo-random functions, encryption and digital signature schemes. Hofheinz *et al.* [91]

Table 4.1. Overview of theoretical code obfuscation results

Weak notion		Strong notion	
+	-	+	-
Canetti [34]			
Canetti *et al.* [37]			
	Barak *et al.* [6]		
	Goldwasser and		
Lynn *et al.* [122]	Kalai [76]		
Wee [186]			Wee [186]
		Hohenberger *et al.* [93]	
		Hofheinz *et al.* [91]	

extended this towards approximate obfuscators for similar types of families. On the other hand, it has been pointed out in several papers (e.g., [6, 93]) that the predicate-based definition is too weak to capture any meaningful result, since useful non-black-box information might still leak (a concrete example of this is Theorem 2).

Nevertheless, the impossibility results for obfuscation by Barak *et al.* [6] and Goldwasser *et al.* [76] do not rule out the existence of an obfuscator for a *specific* meaningful (i.e., non-trivial) natural class of functions. Specifically within the context of white-box cryptography. It is conceivable that a definition of obfuscation can be formulated falling somewhere in between the two extremes, which is neither too weak nor too strong, and can be used for proving white-box security of arbitrary cryptographic primitives. This will be our objective in the remainder of this chapter.

4.3 Security Notions

A major area of study in cryptology involves formal security models to assess the security of cryptographic primitives. Formal security models specify how an adversary can interact with (legitimate users of) a cryptosystem, and what should be achieved in order to *break* the cryptosystem. We refer to Dent [55] for an overview of models in provable security, and their respective issues.

We follow the basic principle of various "game-based" approaches [9, 11, 78, 79], where an attack is captured using an interactive game with an adversary. A *challenger* generates all keys in the model, and may respond to queries from the adversary. The game terminates when the adversary terminates (with polynomial time), and a cryptographic primitive is said to be secure (under the specified security notion), when the probability of success (i.e., the adversary meets the

conditions specified to break the system) is small. A *Security Notion* (SN) is a formal description of the security desired from a cryptographic scheme [9, 11], and defines what capabilities the attacker is given and what constitutes a successful attack. Security notions for many types of cryptographic primitives have been proposed, including digital signatures [80], asymmetric encryption schemes [153], and symmetric encryption schemes [8].

For convenience, we make the following assumptions: (1) a security notion is specific to a cryptographic scheme (thus, IND-CPA-AES, and IND-CPA-DES are two different SNs), (2) all interactions with the adversary is done via oracle queries made by the adversary.

Semantic security

A widely accepted notion of security for asymmetric encryption schemes is *semantic security*. For a (probabilistic) encryption scheme to be semantically secure, it must be infeasible for a computationally bounded adversary to derive additional *information* about a plaintext, when given the corresponding ciphertext (and the public key K). This notion was introduced by Goldwasser and Micali [78]. Subsequently, Goldwasser and Micali [79] showed that semantic security follows from *ciphertext indistinguishability*. This allowed the use of the IND-CPA security notion as a more commonly used notion to assess the security of asymmetric encryption schemes.

The IND-CPA security notion for a public-key cryptosystem defines that the adversary has access to the encryption oracle and needs to guess a secret bit. Hence, a public-key cryptosystem is defined to be IND-CPA secure if no adversary with access to an encryption oracle can pick two messages, of equal length, such that it can distinguish (still having encryption-oracle access) between the encryptions of the two. This notion is similar to the *find-then-guess* CPA (FTG-CPA) security by Bellare *et al.* [8] (for symmetric ciphers).

Definition 9 (IND-CPA security) *The IND-CPA game is defined by the following steps:*

- **Setup.** *Challenger C sends the public key K to the adversary A, and privately chooses a bit $b \leftarrow \{0,1\}$.*
- **Challenge.** *A computes two equal length messages (m_0, m_1) and sends these to C, who responds with the encryption $c^* = \varepsilon_K(m_b)$.*
- **Output.** *Within a polynomial-complexity time, A outputs $a \in \{0,1\}$.*

Then,

$$Adv_A^{IND\text{-}CPA} = \left| \Pr[a = b] - \frac{1}{2} \right| .$$

A scheme is said to be ϵ-IND-CPA secure if $Adv_A^{IND\text{-}CPA} \leq \epsilon$ (win condition).

Another commonly used security notion is IND-CCA [153], similar to the FTG-CCA in [8] for symmetric schemes. It differs from IND-CPA, in that an adversary is given access to a decryption oracle (prior to receiving the challenge). This is often denoted as *non-adaptive CCA*. In *adaptive CCA* (IND-CCA2), the adversary is allowed access to the decryption oracle also after receiving the challenge. Obviously, access to the decryption oracle is limited in the sense that decryption of the challenged ciphertext is not allowed.

Left-or-right security

To analyze the security of symmetric key primitives, Goldreich [71] suggested to treat this similarly as in the asymmetric setting. However, to model for example IND-CCA2 attacks, one must provide the adversary a means to encrypt, while this is by definition available in the asymmetric setting (by means of the public key K). This presence of an encryption oracle is one of the reasons why notions of symmetric encryption cannot be treated as a special case of asymmetric encryption. In [8], Bellare *et al.* suggested four notions of security for symmetric encryption. We refer to [8] for a formalization of, and a comparison between the four notions of security.

Let us present the notion of *"left-or-right security"*. This notion specifies two different games, `Game 1`, and `Game 2`, as depicted in Fig. 4.2. Both games are initiated with the same randomly chosen key $k \leftarrow \mathcal{K}$, which remains fixed for the duration of the game. On input (m_1, m_2), `Game` i responds with $\varepsilon_k(m_i)$, the probabilistic encryption of m_i under key k, with random seed r. The goal of the adversary is to distinguish the oracles.

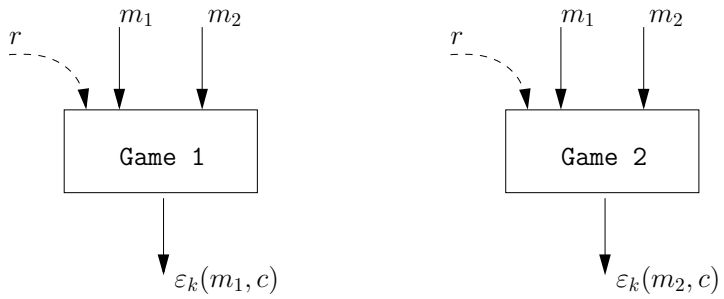

Figure 4.2. Left-or-right security

Definition 10 (Left-or-Right security [8]) *A (probabilistic) symmetric encryption scheme $(\varepsilon, D, \mathcal{K})$ is said to be $(t, q, u; \epsilon)$-secure in the left-or-right sense, if for any adversary A that runs in at most time t and asks at most q queries for a total of at most μ bits, holds that*

$$Adv_A^{lr} = \left| \begin{array}{l} \Pr\left[k \leftarrow \mathcal{K} : A^{\varepsilon_k(left(.,.))} = 1\right] - \\ \Pr\left[k \leftarrow \mathcal{K} : A^{\varepsilon_k(right(.,.))} = 1\right] \end{array} \right| \leq \epsilon\,,$$

where the probabilities are taken over the coin tosses of A, ε_k, and over the distribution of $\mathcal{K}$.

White-box security

The (black-box) security of any primitive is captured using a security notion where the adversary is given black-box access to some functionality (e.g., encryption), and a white-box implementation can be required to satisfy that security notion when the adversary is given access to a *white-boxed* version of the functionality. We would like an obfuscation to ensure that all the security notions are satisfied in the white-box variant when they are satisfied in the black-box variant. However, from prior work, it is not fully clear if any of the existing definitions of obfuscation can be used to achieve such a goal. Hence, a natural question is:

Given an obfuscator satisfying the virtual black-box property for a program P, and some security notion that is satisfied when the adversary is given black-box access to P, can it be proved that the security notion remains satisfied when the adversary is also given access to the obfuscated program $\mathcal{O}(P)$?

4.4 Our Contribution

The contributions in this chapter are two-fold. First, in Sect. 4.7, we develop the foundations of white-box cryptography by formalizing the notion of security that cryptographic primitives must satisfy. In order to do this, we define the *white-box property* (WBP) which captures the security of an obfuscated program with respect to some given security notion. If satisfied, the WBP implies that the obfuscation does not leak any useful information under that security notion, even though it may leak useful information under a different security notion.

Second, in Sect. 4.8, we present some (im)possibility results about the reduction between WBP and obfuscation and answer the above natural question in the negative – we show that under any definition of obfuscation, the answer is, in general, no. In other words, we show that for most programs P, there cannot exist an obfuscator that satisfies the WBP for all security notions in which P might be present. We also show impossibility results for the composition of white-box

implementations. On the positive side, we show that under reasonable computational assumptions, there exists an obfuscator that satisfies the WBP with respect to a meaningful security notion for a meaningful cryptographic primitive. We also show that there exist obfuscators that satisfy the WBP with respect to *every* security notion for a (contrived) non-learnable, but approximate learnable family.

To understand these results, it is important to note that obfuscation and white-box cryptography are two distinct concepts and should not be confused with each other. The soundness of obfuscation is captured by a Virtual Black-Box Property, which is defined with respect to a program alone [6, 76, 91, 93, 186], while WBC is captured using a WBP, which is always defined with respect to a program and a security notion.

4.5 Preliminaries

The following definition settles some terminology that we need to formalize our notion of white-box cryptography.

Definition 11 *In the following, unless otherwise stated, a TM is assumed to have one input tape.*

1. **(Equality of TMs.)** *$X, Y \in \mathbb{TM}$ are equal (written $X = Y$) if $\forall a : X(a) = Y(a)$.*

2. **(Polynomial TM.)** *$X \in \mathbb{TM}$ is a Polynomial TM (PTM) if there exists $p \in \mathbb{P}$ such that $\forall a : X(a)$ halts in at most $p(|a|)$ steps. Let $\mathbb{PTM}$ be the set of al PTMs.*

3. **(PPT Algorithms.)** *A PPT algorithm (such as an adversary or an obfuscator) is a PTM with an unknown source of randomness input via an additional random tape. We denote the set of PPT algorithms by $\mathbb{PPT}$. The running time of a PPT algorithm must be polynomial in the length of the known inputs.*

4. **(TM Family.)** *A TM Family (TMF) is a TM having two input tapes: a* key tape *and a* standard input tape. *We denote by $\mathbb{TMF}$ the set of all TMFs. Let $Q \in \mathbb{TMF}$. Then:*

   (a) *The symbol $Q[q]$ indicates that the key tape of Q contains the string q.*

   (b) *We denote by $\mathcal{K}_Q$ the key-space (valid strings for the key tape) of Q.*

   (c) *Let $q \in \mathcal{K}_Q$. In our model, the input-space (valid strings for the standard input tape) of $Q[q]$ is fully defined by the parameter $|q|$. We*

denote this space by $\mathcal{I}_{Q,|q|}$. *Furthermore there must exist a polynomial* $\mathbf{P}_Q \in \mathbb{P}$ *such that:*

$$\forall q \in \mathcal{K}_Q, \forall x \in \mathcal{I}_{Q,|q|} : |x| = \mathbf{P}_Q(|q|)\,.$$

All TMFs considered in this work are deterministic. That is, for any $(Q, q) \in \mathbb{TMF} \times \mathcal{K}_Q$ *the output of* $Q[q]$ *is fully defined by the input. For modeling probabilistic algorithms using TMFs we will assume that randomness is supplied as part of* q *and/or the input.*

5. **(Polynomial TM Family.)** $Q \in \mathbb{TMF}$ *is a* Polynomial *TMF (PTMF) if there exists* $p \in \mathbb{P}$ *such that* $\forall q \in \mathcal{K}_Q, \forall a \in \mathcal{I}_{Q,|q|} : Q[q](a)$ *halts in at most* $p(|q|)$ *steps. We denote the set of all PTMFs by* $\mathbb{PTMF}$.

6. **(Learnable Family.)** $Q \in \mathbb{TMF}$ *is learnable if* $\exists (L, p) \in \mathbb{PPT} \times \mathbb{P}$ *s.t.*

$$\forall k : \Pr\left[q \xleftarrow{R} \{0,1\}^k \cap \mathcal{K}_Q; X \leftarrow L^{Q[q]}(1^{|q|}, Q) : X = Q[q]\right] \geq 1/p(k)$$

(the probability taken over the coin tosses of L*) and:*

   (a) $\forall a$: *if* $Q[q](a)$ *halts after* t *steps then* $X(a)$ *halts after at most* $p(t)$ *steps.*[2]

   (b) $|X| \leq p(|q|)$.

Informally, a function Q *is learnable when, by means of a limited number of queries to its functionality, an equivalent function* X *can be constructed.* L *is called the learner for* Q*, and* $\mathbb{LF}$ *is the set of all learnable families.*

7. **(Approximate Learnable Family.)** $Q \in \mathbb{TMF}$ *is approximate learnable if* $\exists (L, p) \in \mathbb{PPT} \times \mathbb{P}$ *s.t.*

$$\forall k : \Pr\left[q \xleftarrow{R} \{0,1\}^k \cap \mathcal{K}_Q; a \xleftarrow{R} \mathcal{I}_{Q,k}; X \leftarrow L^{Q[q]}(1^{|q|}, Q) : X(a) = Q[q](a)\right] \geq 1/p(k)$$

(the probability taken over the coin tosses of L*), and:*

   (a) $\forall a$: *if* $Q[q](a)$ *halts after* t *steps then* $X(a)$ *halts after at most* $p(t)$ *steps.*

   (b) $|X| \leq p(|q|)$.

[2] This condition is to prevent an exponential time learner from becoming polynomial time by hard-wiring the learning algorithm and queries/responses inside X.

An approximate learnable family is a relaxed notion of a learnable family, in the sense that the equality should only hold for a randomly selected input a, not for the entire input space. For example, point functions are an approximate learnable family, but not a learnable family.

We denote the set of all approximate learnable families by $\mathbb{ALF}$.

Lemma 1 captures that if two families behave identically in a black-box way, and the corresponding keys are of equivalent size, then both families are either learnable or non-learnable. This lemma will be used in the proofs of our results, such as Theorem 4.

Lemma 1 *If* $Q_1 \in \mathbb{PTMF} \backslash (\mathbb{A})\mathbb{LF}$*, then the following holds:*

$$\Big[\exists (Q_2, p) \in \mathbb{PTMF} \times \mathbb{P}, \forall q_1 \in \mathcal{K}_{Q_1}, \exists q_2 \in \mathcal{K}_{Q_2} : Q_1[q_1] = Q_2[q_2] \wedge |q_2| \leq p(|q_1|)\Big] \rightarrow Q_2 \notin (\mathbb{A})\mathbb{LF}.$$

The symbol $(\mathbb{A})$ *indicates that* $\mathbb{A}$ *is optional in the above statement.*

Proof: Assume for contradiction that for any given $Q_1 \in \mathbb{PTMF}$ that is not learnable, there exists some $(Q_2, p) \in \mathbb{PTMF} \times poly$ such that the left-hand side of the above implication is satisfied but the right-hand side is not. Let L_1 and L_2 be the learners for Q_1 and Q_2 respectively. L_1 runs L_2 using its own oracle to answer L_2's queries. If Q_2 is learnable, then L_2 will output $X_2 = Q_2[q_2]$ with non-negligible (in $|q_2|$) probability in a polynomial (of $|q_2|$) number of steps, which is a polynomial function of $|q_1|$ by assumption, a contradiction. □

4.6 Obfuscators

Informally, an obfuscator $\mathcal{O}$ is a probabilistic compiler that transforms a program P into $\mathcal{O}(P)$, a functionally equivalent implementation of P which hides certain characteristics of P.

In this work, we only consider obfuscation of PTMFs with a uniformly selected key, and not of a single PTM. Denote by Q a family of cryptographic primitives (a PTMF), for which the description is publicly known. Denote by $Q[q]$ a primitive instantiated with secret key q selected from some distribution on the key-space $\mathcal{I}_{Q,|q|}$. As is common in cryptography, we define the functionality of the obfuscator using a *correctness* property and the security using a *soundness* property.

4.6.1 Obfuscator (Correctness)

Definition 12 *A randomized algorithm* $\mathcal{O} : \mathbb{PTMF} \times \{0,1\}^* \mapsto \mathbb{TM}$ *is an (efficient) obfuscator for* $Q \in \mathbb{PTMF}$ *if it satisfies* correctness *defined using the following properties:*

1. ***Approximate functionality:***

$$\forall q \in \mathcal{K}_Q, \forall a \in \mathcal{I}_{Q,|q|} : \Pr\left[\mathcal{O}(Q,q)(a) \neq Q[q](a)\right] \leq neg(|q|)\,,$$

   the probability taken over the coin tosses of $\mathcal{O}$*.*

2. ***Polynomial slowdown and expansion:*** *There exists* $p \in \mathbb{P}$ *s.t.*

$$\forall q \in \mathcal{K}_Q : |\mathcal{O}(Q,q)| \leq p(|q|)\,,$$

   and $\forall a$*, if* $Q[q](a)$ *halts in* t *steps then* $\mathcal{O}(Q,q)(a)$ *halts in at most* $p(t)$ *steps.*

For now, we consider the functionality of $\mathcal{Q}$ only in a deterministic sense and do not explicitly consider the notion of obfuscation of "probabilistic functions" (used, for example, in [91, 93]). However, our negative results (presented in Sect. 4.8.1) also apply to probabilistic functions using an appropriately defined notion of *probabilistic PTMFs (PPTMFs)* (and a corresponding notion of approximate functionality for PPTMFs). This aspect will be further discussed in Sect. 4.9.

4.6.2 Obfuscator (Soundness)

Over recent years, several definitions of soundness have been proposed, all based on some sort of *Virtual Black-Box Property (VBBP)* [6, 91, 93, 122, 186] (see Sect. 4.2). Let $Q \in \mathbb{PTMF}$ and let $q \in \{0,1\}^*$. Loosely speaking, the VBBP requires that whatever information about q a PPT adversary computes given the obfuscation $\mathcal{O}(Q,q)$, a PPT simulator could also have computed using only black-box access to $Q[q]$. All existing notions of VBBP can be classified into one of two broad categories. At one extreme (the weakest) are the predicate-based definitions, where the adversary and the simulator are required to compute some predicate of q. At the other extreme (the strongest) are definitions based on computational indistinguishability, where the simulator is required to output something that is indistinguishable from $\mathcal{O}(Q,q)$. We define these two notions below. Our definitions are based on those of [76], where an auxiliary input is also considered.

Definition 13 *An obfuscator* $\mathcal{O}$ *for* $Q \in \mathbb{PTMF}$ *satisfies* soundness *for* Q *if at least one of the properties given below is satisfied.*

1. ***Predicate Virtual black-box property (PVBBP):*** *Let π be any efficiently verifiable predicate on $\mathcal{K}_Q$. $\mathcal{O}$ satisfies PVBBP for Q if*

$$\forall (A,p) \in \mathbb{PPT} \times \mathbb{P}, \exists (S,k') \in \mathbb{PPT} \times \mathbb{N}, \forall k > k' : Adv^{pvbbp}_{A,S,\mathcal{O},Q}(k) \leq negl(k)\,,$$

where the advantage of the adversary $Adv^{pvbbp}_{A,S,\mathcal{O},Q}(k)$ is defined as

$$\max_{\pi} \max_{z \in \{0,1\}^{p(k)}} \left| \begin{array}{c} \Pr[q \xleftarrow{R} \{0,1\}^k \cap \mathcal{K}_Q : A^{Q[q]}(1^k, \mathcal{O}(Q,q), z) = \pi(q)] \\ -\Pr[q \xleftarrow{R} \{0,1\}^k \cap \mathcal{K}_Q : S^{Q[q]}(1^k, z) = \pi(q)] \end{array} \right|,$$

the probability taken over the coin tosses of $\mathcal{O}, A, S$.[3]

2. ***Computational Indistinguishability (IND):*** *$\mathcal{O}$ satisfies IND for Q if*

$$\forall (A,p) \in \mathbb{PPT} \times \mathbb{P}, \exists (S,k') \in \mathbb{PPT} \times \mathbb{N}, \forall k > k' : Adv^{ind}_{A,S,\mathcal{O},Q}(k) \leq negl(k)\,,$$

where the advantage of the adversary $Adv^{ind}_{A,S,\mathcal{O},Q}(k)$ is defined as

$$\max_{z \in \{0,1\}^{p(k)}} \left| \begin{array}{c} \Pr[q \xleftarrow{R} \{0,1\}^k \cap \mathcal{K}_Q : A^{Q[q]}(1^k, \mathcal{O}(Q,q), z) = 1] \\ -\Pr[q \xleftarrow{R} \{0,1\}^k \cap \mathcal{K}_Q : A^{Q[q]}(1^k, S^{Q[q]}(1^k, z), z) = 1] \end{array} \right|,$$

the probability taken over the coin tosses of $\mathcal{O}, A, S$.

Depending on the property satisfied, we call it IND-soundness or PVBBP-soundness (note that the former implies the latter).

It has been noted (informally) in several papers (e.g., [6, 93]) that the PVBBP is too weak for practical purposes. Furthermore, it has been noted that the IND-soundness is too strong to be satisfied in practice [93, 186]. This can be captured in a formal way as follows.

Proposition 2 *If there exists an obfuscator satisfying IND-soundness for some $Q \in \mathbb{PTMF}$ then $Q \in \mathbb{ALF}$.*

This result does not hold if the definition of approximate functionality in correctness is extended to probabilistic functions. See Sect. 4.9 for details.

[3] The definition of PVBBP given here is slightly weaker than the one used in [6] because they require this property to hold for every q, while we require it to hold only for uniformly selected q.

4.7 White-Box Cryptography

In this section, we formalize the notion of WBC by defining a *white-box property (WBP)*. A key concept in our model are security notions, which capture the capabilities of the adversary, and what constitutes a successful attack (see Sect. 4.3). Definition 14 presents a formal definition of a security notion.

Definition 14 *A **Security Notion** (SN) is a 5-tuple* $(n, p_{in}, \mathbf{Q}, \mathsf{Extr}, \mathsf{Win}) \in \mathbb{N} \times \mathbb{P} \times \mathbb{PTMF}^n \times \mathbb{PTM} \times \mathbb{PTM}$ *such that:*

- $\mathbf{Q} = (Q_1, Q_2, \ldots, Q_n) \in \mathbb{PTMF}^n$ *is an array of (the descriptions of) n PTMFs.*
- Extr *and* Win *are (the description of) PTMs of the type* $\{0,1\}^{p_{in}(k)} \mapsto \times_{i=1}^{n} \mathcal{K}_{Q_i}$ *and* $\{0,1\}^* \mapsto \{0,1\}$ *respectively.*

Denote by $\mathbb{SN}$ *the set of all security notions. For any* $sn = (n, p_{in}, \mathbf{Q}, \mathsf{Extr}, \mathsf{Win}) \in \mathbb{SN}$ *and any* $Q \in \mathbb{PTMF}$*, we say* $Q \in sn$ *if* $Q \in \mathbf{Q}$*.*

The Win predicate captures the requirements and objectives of the security notions, in the sense that it verifies if the adversary did not exceed his capabilities, and has achieved the objectives. The Extr function describes the generation of the keys by the challenger. In the following sections, we introduce the black-box game, and the white-box game, which describe the games that the challenger and the adversary play, in respectively the black-box attack context and the white-box attack context.

4.7.1 Black-Box Game

Definition 15 describes the black-box game, which captures the success probability of an adversary A to analyze a set of cryptographic primitives $\mathbf{Q}$ under a security notion $sn \in \mathbb{SN}$ when given oracle access to these primitives.

Definition 15 *A **Black-box Game** is a TM describing an interactive protocol with the adversary* $A \in \mathbb{PPT}$*. It has a standard structure given below. It takes as input a tuple* $(1^k, sn, r)$*, where* $sn = (n, p_{in}, \mathbf{Q}, \mathsf{Extr}, \mathsf{Win}) \in \mathbb{SN}$ *is a security notion, and* $r \in \{0,1\}^{p_{in}(k)}$ *is a string (representing randomness supplied to the game). It outputs* 0 *or* 1*. The black-box game is given in Algorithm 5 (*GameBB_A*).*

$\mathsf{Queries}$ *is a set representing oracle queries made by A during the game. Each element j of this set is an ordered tuple of the type*

$$(\mathbf{t}_j, \mathbf{i}_j, \mathbf{in}_j, \mathbf{out}_j) \in \mathbb{N} \times \{1, 2, \ldots, n\} \times \{0,1\}^* \times \{0,1\}^*,$$

Algorithm 5: $\mathsf{GameBB}_A(1^k, sn, r)$

input : $1^k, sn, r$
Parse sn as $(n, p_{in}, \mathbf{Q}, \mathsf{Extr}, \mathsf{Win})$
Parse $\mathbf{Q}$ as $(Q_1, Q_2, \ldots, Q_n)$
/* extract keys for each family */
$(q_1, q_2, \ldots q_n) \leftarrow \mathsf{Extr}(r)$
/* interact with adversary */
$s \leftarrow A^{Q_1[q_1], Q_2[q_2], \ldots, Q_n[q_n]}(1^k, sn)$
/* decide if adversary won */
$\mathsf{result} \leftarrow \mathsf{Win}(r, \mathsf{Queries}, s)$
output: result

indicating respectively, the time, oracle number, input, and the output of each query.[4] *The game has two important rules: (1) at any instant A can query at most one oracle, and (2) each query by the adversary takes one unit time irrespective of the amount of computation involved. Denote by* $AdvBB_A^{sn}(k)$ *the black-box advantage of an adversary A, defined as*

$$\Pr\left[r \xleftarrow{R} \{0,1\}^{p_{in}(k)} : \mathsf{GameBB}_A(1^k, sn, r) = 1\right],$$

the probability taken over the coin tosses of A.

An example game for the IND-CCA2 security notion

Let $\mathcal{E} = (G, E, D)$ be a symmetric encryption scheme. The key generation algorithm G takes in as input the security parameter (1^k) and a k-bit random string. It outputs a k-bit symmetric key K. As an example, we describe the *Indistinguishability under Adaptive Chosen Ciphertext Attack* (IND-CCA2) of $\mathcal{E}$ using security notion *ind-cca2*-$\mathcal{E}$ $= (3, p_{in}, \mathbf{Q}, \mathsf{Extr}, \mathsf{Win})$, with $p_{in}(k) = 2k + 1$ and $\mathbf{Q} = (\mathbf{E}, \mathbf{D}, \mathbf{C})$ in Algorithm 6).

$\mathbf{E} \in_{obf}$ *ind-cca2*-$\mathcal{E}$ since the Win predicate does not consider the queries made to the encryption oracle $\mathbf{E}[K]$.

Discussion on Obfuscatability

Observe the IND-CCA2 game described in Algorithm 6. The adversary has oracle access to the encryption and decryption functionality $\mathbf{E}[K]$ and $\mathbf{D}[K]$

[4]Note that the last element of this tuple (the output) is redundant because it is efficiently computable from the input alone. However, including it makes some definitions simpler (for instance IND-CCA2 security).

Algorithm 6: Security Notion *ind-cca2-*$\mathcal{E}$

input : $1^k, r$
Function $\mathbf{E}[K](\textbf{\textit{Input}}\ \langle \alpha, m \rangle)$ **{**
  output $E(K, \alpha, m)$
}
Function $\mathbf{D}[K](\textbf{\textit{Input}}\ c)$ **{**
  output $D(K, c)$
}
Function $\mathbf{C}[\langle b, K, \beta \rangle](\textbf{\textit{Input}}\ \langle m_0, m_1 \rangle)$ **{**
  if $(|m_0| = |m_1|)$ **then** output $E(K, \beta, m_b)$ **else** output 0
}
Function Extr(***Input*** r) **{**
  parse r as $\langle \gamma, \beta, b \rangle$;
  $K \leftarrow G(1^{|\gamma|}, \gamma)$;
  output $K, K, \langle b, K, \beta \rangle$
}
Function Win(***Input*** $(r, \mathsf{Queries}, s)$) **{**
  Parse r as (q, x, a);
  if *((At most one query to* $\mathbf{C}[\langle b, K, \beta \rangle]$*) AND No query to* $\mathbf{D}[K]$ *on output of* $\mathbf{C}[\langle b, K, \beta \rangle]$ *after query to* $\mathbf{C}[\langle b, K, \beta \rangle]$*) AND* $(s = b)$**) then Output**: 1
  else Output: 0
}

respectively, with K the instantiated secret key. The adversary wins if he can guess the bit b correctly, without querying $\mathbf{D}[K]$ with the challenge ciphertext m_b.

Then, $\mathbf{D}[K]$ cannot be obfuscated, since this would render the corresponding asymmetric scheme $\mathbf{E}$ insecure under IND-CCA2: once the adversary has obtained an executable implementation of $\mathbf{D}[K]$, the challenger cannot prevent the adversary to query the decryption function on the challenge ciphertext (the adversary does not even have to 'break' any obfuscation in order to do this). On the other hand $\mathbf{E}[K]$ is a perfect candidate for obfuscation since the winning condition does not depend on what was queried to $\mathbf{E}[K]$.

We introduce the concept of 'obfuscatability' in Definition 16 to capture when a cryptographic primitive (i.e., family) is a suitable candidate for obfuscation. Intuitively, Definition 16 says that a family Q_i is a possible candidate for obfuscation (within the context of a security notion), when the winning condition is not affected by the number of queries to the functionality of Q_i.

Definition 16 *For any $sn \in \mathbb{SN}$ and $Q_i \in sn$, the family Q is denoted **Obfuscatable** if the output of* Win *is invariant with respect to the entries of* Queries *corresponding to oracle $Q_i[q_i]$ in a black-box game. Formally, for any PTMF $Q_i \in sn$, define* $\mathsf{Queries}_i$ *to be the following set:*

$$\{(\mathbf{t}_j, \mathbf{i}_j, \mathbf{in}_j, \mathbf{out}_j) | (\mathbf{t}_j, \mathbf{i}_j, \mathbf{in}_j, \mathbf{out}_j) \in \mathsf{Queries} \wedge \mathbf{i}_j \neq i\}$$

Q_i is obfuscatable *in sn (written $Q_i \in_{obf} sn$) if*

$$\forall r, \mathsf{Queries}, s : \mathsf{Win}(r, \mathsf{Queries}, s) = \mathsf{Win}(r, \mathsf{Queries}_i, s).$$

Note: If $Q \notin sn$ then $Q \notin_{obf} sn$.

We claim that a meaningful notion of white-box security cannot be obtained for a family under a security notion in which it is not obfuscatable. For instance, white-boxing the decryption function of a symmetric encryption scheme, or the 'signing' function of a MAC scheme under any standard security notion is not meaningful (however, it is possible to construct specialized/contrived security notions in which this becomes meaningful).

4.7.2 White-Box Game

In Definition 17, we extend the black-box game to a white-box game, which captures the success probability of an adversary A to analyze a set of cryptographic primitives $\mathbf{Q}$ under a security notion $sn \in \mathbb{SN}$ when given oracle access to these primitives, and given full access to the primitive Q_i that is obfuscated by $\mathcal{O}$.

Definition 17 *A **White-Box Game** is defined for the tuple $(A, \mathcal{O}, Q_i)$ by extending the black-box game. For a security notion $sn = (n, p_{in}, \mathbf{Q}, \mathsf{Extr}, \mathsf{Win}) \in \mathbb{SN}$ with $(\mathbf{Q} = (Q_1, Q_2, \ldots, Q_n)$, assume that $Q_i \in_{obf} sn$ $(1 \leq i \leq n)$ and that $\mathcal{O} \in \mathbb{PPT}$ is an obfuscator for Q_i. The white-box game is given in Algorithm 7 ($\mathsf{GameWB}_{A,\mathcal{O},Q_i}$).*

Denote by $AdvWB^{sn}_{A,\mathcal{O},Q_i}(k)$, the advantage of an adversary in the white-box game, defined as

$$\Pr\left[r \xleftarrow{R} \{0,1\}^{p_{in}(k)} : \mathsf{GameWB}_{A,\mathcal{O},Q_i}(1^k, sn, r) = 1\right],$$

the probability taken over the coin tosses of A.

Definition 18 *Let $\mathcal{O}$ be an obfuscator for $Q \in \mathbb{PTMF}$ and let $sn \in \mathbb{SN}$ such that $Q \in sn$. The **White-box Advantage** (WBA) of $\mathcal{O}$ for (Q, sn), $AdvWB^{sn}_{\mathcal{O},Q}(k)$, is defined as*

$$\left| \max_{A \in \mathbb{PPT}} AdvWB^{sn}_{A,\mathcal{O},Q}(k) - \max_{A \in \mathbb{PPT}} AdvBB^{sn}_{A}(k) \right|.$$

Algorithm 7: $\mathsf{GameWB}_{A,\mathcal{O},Q_i}(1^k, sn, r)$

input : $1^k, sn, r$
Parse sn as $(n, p_{in}, \mathbf{Q}, \mathsf{Extr}, \mathsf{Win})$
Parse $\mathbf{Q}$ as $(Q_1, Q_2, \ldots, Q_n)$
`/* extract keys for each family */`
$(q_1, q_2, \ldots q_n) \leftarrow \mathsf{Extr}(r)$
`/* interact with adversary */`
$s \leftarrow A^{Q_1[q_1], Q_2[q_2], \ldots, Q_n[q_n]}(1^k, sn, i, \mathcal{O}(Q_i, q_i))$
`/* decide if adversary won */`
$\mathsf{result} \leftarrow \mathsf{Win}(r, \mathsf{Queries}, s)$
output: result

The WBA serves as a measure of useful information leakage by an obfuscation.

The term 'useful information' within the context of the security notion is any information that aids the adversary in conducting a successful attack.

Definition 19 *Let $\mathcal{O}$ be an obfuscator for $Q \in \mathbb{PTMF}$ and let $sn \in \mathbb{SN}$ such that $Q \in sn$. $\mathcal{O}$ satisfies the **White-box Property (WBP)** for (Q, sn) if*

$$AdvWB^{sn}_{\mathcal{O},Q}(k) \leq neg(|k|)\,.$$

This captures the notion of obfuscation, in the sense that the best adversary in a white-box setting is not able to extract significantly more useful information than the best adversary in a black-box setting.

Definition 20 *Let $\mathcal{O}$ be an obfuscator for $Q \in \mathbb{PTMF}$. $\mathcal{O}$ satisfies the **Universal White-box Property (UWBP)** for Q if for every $sn \in \mathbb{SN}$ with $Q \in_{obf} sn$, $\mathcal{O}$ satisfies WBP for (Q, sn).*

Definition 19 and 20 give us a formal, sensible meaning of what the objective of white-box cryptography is. The WBP, if satisfied would imply that, given a cryptographic primitive that is secure in the black-box sense, its white-box implementation also remains secure with respect to the desired security notion. In the next section, we investigate what can be achieved within the context of our model.

4.8 (Im)possibility Results

In this section we give some useful relationships between obfuscators, WBP and UWBP.

4.8.1 Negative Results

Barak *et al.* [6] gave several impossibility results on obfuscation, some of them quite strong (for instance they present an unobfuscatable encryption scheme). Our negative results are even stronger than theirs. To put our main negative result in context with that of [6], we first mention their result using our notation.

Proposition 3 *(Barak* et al. *[6]) There exists a pair* $(Q, sn) \in \mathbb{PTMF} \times \mathbb{SN}$ *with* $Q \in_{obf} sn$ *such that every obfuscator for* Q *fails to satisfy WBP for* (Q, sn)*.*

In other words, there cannot exist an obfuscator that satisfies UWBP for every $Q \in \mathbb{PTMF}$. However, their results do not rule out an obfuscator that satisfies the UWBP for some useful family Q. We show that even this is not possible unless Q is at least approximate learnable.

No UWBP for "Interesting" Families

Obfuscators that satisfy the UWBP for "interesting" families do not exist. That is, in Theorem 2, we show that any non-approximately-learnable family, there exists a security notion that cannot be satisfied when an adversary has white-box access to the white-box implementation.

Theorem 2 *For every family* $Q \in \mathbb{PTMF} \backslash \mathbb{ALF}$*, there exists a (contrived)* $sn \in \mathbb{SN}$ *such that* $Q \in_{obf} sn$ *but every obfuscator for* Q *fails to satisfy the WBP for* (Q, sn)*.*

Proof: Let $Q \in \mathbb{PTMF} \backslash \mathbb{ALF}$. Consider the security notion *guess-x*, defined as $(2, p_{in}, (Q, Q_1), \mathsf{Extr}, \mathsf{Win}) \in \mathbb{SN}$, with $p_{in}(k) = 2k + \mathbf{P}_Q(k)$, described in Algorithm 8.

Observe that $Q \in_{obf}$ *guess-x*. Since $Q \notin \mathbb{ALF}$, therefore by virtue of Definitions 11.7 and 12.1, for sufficiently large k, the following inequalities are guaranteed to hold:

$$\forall A \in \mathbb{PPT} : 0 \leq \mathrm{AdvBB}_{A}^{\textit{guess-x}}(k) < \alpha(k)\,,$$

$$\exists A \in \mathbb{PPT} : 1 \geq \mathrm{AdvWB}_{A,\mathcal{O},Q}^{\textit{guess-x}}(k) \geq 1 - \beta(k)\,,$$

where α, β are negligible functions. Hence, we have that

$$\mathrm{AdvWB}_{\mathcal{O},Q}^{\textit{guess-x}}(k) > 1 - \alpha(k) - \beta(k)\,,$$

Algorithm 8: Q_1, Extr and Win for *guess-x*.

Function $Q_1[q_1]$ (***Input*** Y_1) **{**
/* Assume: $Y_1 \in \mathbb{PTM}$ */
Parse q_1 as (q, x, a)
if $(Y_1(a) = Q[q](a))$ **then** output x **else** output 0
/* Q is used as a black-box and let $Q[q]$ halt in t steps */
/* Y_1 is halted after $p(t)$ steps for some (large) $p \in \mathbb{P}$ */
}
Function Extr(***Input*** r) **{**
Parse r as (q, x, a)
/* Assume that $q \in \{0,1\}^k$, $x \in \{0,1\}^k$, and $a \in \{0,1\}^{\mathbf{P}_Q(k)}$ */
/* Without loss of generality, we can assume that */
/* $q \in \mathcal{K}_Q$ and $a \in \mathcal{I}_{Q,k}$ */
set $q_1 \leftarrow (q, x, a)$
output q, q_1
}
Function Win(***Input*** $(r, \mathsf{Queries}, s)$) **{**
Parse r as (q, x, a)
if $(s \neq x) \vee$ (*more than one query to* $Q_1[q_1]$) **then** output 0 **else**
output 1
}

which is non-negligible in k. This proves the theorem. □

Intuitively, for a given non-approximately learnable family Q, the proof of Theorem 2 constructs a special (non-learnable) family of functions Q_1, and defines a specific security notion. A member $Q_1[q_1]$ of this family Q_1 accepts as input a function Y_1, and will output its 'secret' (x) when Y_1 and $Q[q]$ are equivalent for some input a. The security notion specifies that the adversary will win if he can obtain this value x with at most one query to $Q_1[q_1]$.

Since Q is a family of non-learnable functions, a black-box adversary has to guess the value a correctly in order to construct a function that is equivalent to $Q[q]$ in a. Or the adversary has to guess the value x in order to win the game. However, since x and a are randomly selected from a uniform distribution, a black-box adversary has only negligible chance to win the game (black-box game).

An adversary that has the implementation of $Q[q]$ on the other hand, can pass this as an input string to the functionality of $Q_1[q_1]$ in order to recover the value of x (one query to $Q_1[q_1]$ will suffice) and win the game (white-box game). Hence, a white-box adversary has significant advantage to a black-box adversary by having access to the implementation of $Q[q]$, no matter how obfuscated.

Discussion on Impossibility Result

Our result is different from the impossibility result of Barak *et al.*, in the sense that for *any* family $Q \in \mathbb{PTMF} \backslash \mathbb{ALF}$, we can construct a game (security notion) in which $Q[q]$ cannot be obfuscated. Therefore, this result directly applies to 'useful' families of functions such as pseudorandom functions, encryption schemes and MAC algorithms. In contrast, Barak *et al.* construct a *specific* family $Q \in \mathbb{PTMF} \backslash \mathbb{ALF}$ for which no obfuscator can exist, thereby ruling out the existance of a generic obfuscator. Hence our result is even stronger than theirs. The underlying idea of using a function that leaks some secret x is similar.

Our result applies because the approximate functionality requirement for obfuscators (Definition. 12) is defined for deterministic PTMFs only. Extended definitions of probabilistic families can be considered (such as in [91, 93]), and a similar result can be achieved. This aspect is discussed in Sect. 4.9.

Although we define $\mathbb{ALF}$ to be the set of families which can be approximately-learned with a non-negligible advantage (which is quite broad), we note that the above result can be further strengthened by narrowing down the definition of $\mathbb{ALF}$ to only families that can be approximately-learned with an overwhelming advantage.

Our next result deals with multiple obfuscations.

Simultaneous Obfuscation May Be Insecure

A desired property is the composition of obfuscations. When two implementations are securely obfuscated, one would desire that the combination of the two remains secure, as this opens perspectives to many practical applications. Wee [186] and Canetti *et al.* [35] have investigated this question for point functions, while Lynn *et al.* [122] have found a negative answer to this question for generic programs. In Definition 21, we capture the concept of composition within the context of white-box cryptography. In Theorem 3, we show that simultaneous white-boxing of two families may be insecure even if white-boxing each family is secure.

Definition 21 *(Multiple obfuscations) Let $sn \in \mathbb{SN}$ be a security notion and let $Q_i, Q_j \in_{obf} sn$ for some $1 \leq i, j \leq n$. Let $\mathcal{O}$ be an obfuscator for Q_i, Q_j. We extend the white-box game $\mathsf{GameWB}^{sn}_{A,\mathcal{O},Q_i}$ of Definition 17 by defining a new game $\mathsf{GameWB1}^{sn}_{A,\mathcal{O},Q_i,Q_j}$ in which A gets as input the tuple $(1^k, sn, i, j, \mathcal{O}(Q_i, q_i), \mathcal{O}(Q_j, q_j))$. Denote the advantage of an adversary for this new game as $AdvWB^{sn}_{A,\mathcal{O},Q_i,Q_j}(k)$, defined as*

$$\Pr\left[r \xleftarrow{R} \{0,1\}^{p_{in}(k)} : \mathsf{GameWB1}^{sn}_{A,\mathcal{O},Q_i,Q_j}(1^k, r) = 1\right],$$

the probability taken over the coin tosses of $A, \mathcal{O}$. *The obfuscator* $\mathcal{O}$ *satisfies WBP for* $((Q_i, Q_j), sn)$ *if*

$$\left| \max_{A \in \mathbb{PPT}} AdvWB^{sn}_{A,\mathcal{O},Q_i,Q_j}(k) - \max_{A \in \mathbb{PPT}} AdvBB^{sn}_{A}(k) \right|$$

is negligible in $|k|$.

Theorem 3 *Let* $Q_i, Q_j \in \mathbb{PTMF} \backslash \mathbb{ALF}$. *Then there exists a* $sn \in \mathbb{SN}$ *with* $Q_i, Q_j \in_{obf} sn$ *such that even if there exists an obfuscator for* Q_1, Q_2 *satisfying WBP for* (Q_i, sn) *and* (Q_j, sn), *every obfuscator fails to satisfy WBP for* $((Q_i, Q_j), sn)$

The proof is similar to the proof of Theorem 2.

4.8.2 Positive Results

Although the above results rule out the possibility of obfuscators satisfying UWBP for most non-trivial families, they do not imply that a meaningful definition of security for white-box cryptography cannot exist. In fact, any asymmetric encryption scheme can be considered as a white-boxed version of the corresponding symmetric scheme (where the encryption key is also secret). We use this observation as a starting point of our first positive result. A similar observation was used in the positive results of [91].

WBP for "Useful" Families

In Theorem 4, we formally state that there exists a non-approximately learnable family, and an obfuscator that satisfies the WBP for that family for a useful security notion.

Theorem 4 *Under reasonable computation assumptions, there exists a tuple* $(\mathcal{O}, Q, sn) \in \mathbb{PPT} \times \mathbb{PTMF} \backslash \mathbb{ALF} \times \mathbb{SN}$ *such that* $Q \in_{obf} sn$ *and* $\mathcal{O}$ *is an obfuscator satisfying WBP for* (Q, sn).

Before we prove this theorem, we describe a primitive known as bilinear pairing in Definition 22, which we require for the construction in the proof of Theorem 4.

Definition 22 *(Bilinear pairing) Let* G_1 *and* G_2 *be two cyclic multiplicative groups both of prime order* w *such that computing discrete logarithms in* G_1 *and* G_2 *is intractable. A bilinear pairing is a map* $\hat{e} : G_1 \times G_1 \mapsto G_2$ *that satisfies the following properties [24, 25, 27]:*

1. Bi-linearity*: $\hat{e}(a^x, b^y) = \hat{e}(a,b)^{xy}\ \forall a, b \in G_1$ and $x, y \in \mathbb{Z}_w$.*

2. Non-degeneracy*: If g is a generator of G_1 then $\hat{e}(g,g)$ is a generator of G_2.*

3. Computability*: The map $\hat{e}$ is efficiently computable.*

Proof: We prove Theorem 4 by construction of an obfuscatable family Q. We will use an encryption scheme based on the BF-IBE scheme [24].
Define a *symmetric* encryption scheme $\mathcal{E} = (G, E, D)$ as follows.

1. *Key Generation (G):* Let $\hat{e} : G_1 \times G_1 \mapsto G_2$ be a bilinear pairing over cyclic multiplicative groups as defined above (such maps are known to exist). Let $|G_1| = |G_2| = w$ (prime) such that $\lfloor \log_2(w) \rfloor = l$. Pick random $g \xleftarrow{R} G_1 \backslash \{1\}$ and define $\mathcal{H} : G_2 \mapsto \{0,1\}^l$ to be a hash function. Pick $x \xleftarrow{R} G_1$ and define $K = \langle \hat{e}, G_1, G_2, w, g, \mathcal{H}, x \rangle$. The encryption/decryption key is K.

2. *Encryption (E):* Parse K as $\langle \hat{e}, G_1, G_2, w, g, \mathcal{H}, x \rangle$. Let $m \in \{0,1\}^l$ be a message and $\alpha \in \mathbb{Z}_w$ be a random string. The encryption family E is defined as

$$\begin{array}{rcl} E[K] : \{0,1\}^l \times \mathbb{Z}_w & \mapsto & \{0,1\}^l \times G_1 \\ (m, \alpha) & \mapsto & (\mathcal{H}(\hat{e}(x^\alpha, g)) \oplus m, g^\alpha)\,. \end{array}$$

3. *Decryption (D):* Parse K as $\langle \hat{e}, G_1, G_2, w, g, \mathcal{H}, x \rangle$. The decryption family D is defined as

$$\begin{array}{rcl} D[K] : \{0,1\}^l \times G_1 & \mapsto & \{0,1\}^l \\ (c_1, c_2) & \mapsto & \mathcal{H}(\hat{e}(c_2, x)) \oplus c_1\,. \end{array}$$

It can be verified that $D[K](E[K](m, \alpha)) = m$ for valid values of (m, α). The scheme can be proven to be CPA secure if $\mathcal{H}$ is a random oracle and w is sufficiently large. We construct an obfuscation of the $E[K]$ oracle that converts $\mathcal{E}$ into a CPA secure *asymmetric* encryption scheme under a computational assumption.

We define the **obfuscator** $\mathcal{O}$ as follows: the input is (E, K).

1. Parse K as $\langle \hat{e}, G_1, G_2, w, g, \mathcal{H}, x \rangle$. Set $y \leftarrow \hat{e}(x, g)$.

2. Set $K' \leftarrow \langle \hat{e}, G_1, G_2, w, g, \mathcal{H}, y \rangle$ and define family F with key K' as:

$$\begin{array}{rcl} F[K'] : \{0,1\}^l \times \mathbb{Z}_w & \mapsto & \{0,1\}^l \times G_1 \\ (m, \alpha) & \mapsto & (\mathcal{H}(y^\alpha) \oplus m, g^\alpha)\,. \end{array}$$

3. Output $F[K']$.

Claim 1 *$\mathcal{O}$ is an efficient obfuscator for E satisfying WBP for (E, sn), where $sn :=$ "IND-CPA security of $\mathcal{E}$", assuming that the* bilinear Diffie-Hellman assumption *[24] holds in (G_1, G_2) and $\mathcal{H}$ is a random oracle.*

Proof: The IND-CPA security notion captures an adversary that can only perform queries to an encryption oracle with arbitrary plaintexts. The objective is to obtain the plaintext corresponding to a challenge ciphertext. See Algorithm 6 for the IND-CCA2 security notion. The IND-CPA security notion is a restricted version of this where the family **D** is absent.

First note that the obfuscator satisfies correctness for E because $F[K'] = E[K]$. The proof of the above claim follows from the security of the BasicPUB encryption scheme of [24]. □

Claim 2 *If $\mathcal{H}$ is a one-way then $E \in \mathbb{PTMF} \backslash \mathbb{ALF}$.*

Proof: Clearly, $F \in \mathbb{PTMF}$ and the following holds:

$$\exists p \in \mathbb{P}, \forall K \in \mathcal{K}_E, \exists K' \in \mathcal{K}_F :$$

$$F[K'] = E[K] \wedge |K'| = |K| + p(|K|).$$

By virtue of Lemma 1, in order to prove that $E \notin \mathbb{ALF}$, it is sufficient to prove that $F \notin \mathbb{ALF}$. Finally, it is trivial to prove that if $\mathcal{H}$ is a one-way function then $F \notin \mathbb{ALF}$.

□

This completes the proof of Theorem 4. □

An interesting observation from Theorem 4 is that even though the obfuscator $\mathcal{O}$ satisfies WBP for (E, sn), it does not satisfy soundness for E (under Definition 13). This indicates that the obfuscation soundness property and WBP are in general independent of each other.

To justify our choice of the particular scheme (instead of RSA/ElGamal) in the above proof, observe that RSA does not enjoy the security notion of IND-CPA, while Encryption in ElGamal is learnable. To see this, consider the ElGamal encryption oracle:

$$\text{ElGamal Enc}(m, \alpha) = (g^\alpha, m \cdot h^\alpha)\,,$$

with g the generator for a multiplicative group G, x the private key, and $h = g^x \in G$ the public key. The ElGamal encryption cannot be obfuscated, since the encryption key is learnable with a single query: $E(1, 1) = (g, h)$. The obfuscator of [93] does not satisfy our definition of approximate functionality, and hence their scheme is unsuitable for the proof. (However, the scheme of [93] is the ideal candidate for an analogous approach on probabilistic functions in Sect. 4.9.)

Discussion on Positive Result

We have presented a symmetric scheme that can be implemented by an obfuscator in such a way that it remains provably CPA secure. However, in the proof of Theorem 4, we have used a primitive that consists of building blocks (pairings) that are typically used in asymmetric cryptography. This can be seen as an admission in the sense that we started the research of white-box cryptography with the obfuscation of symmetric block ciphers such as DES and AES.

However, cryptographic schemes that can be securely implemented (under a CPA security notion) naturally turn into asymmetric primitives, since CPA security is a basic notion. Also, because we attempt to achieve a provable degree of security, we naturally fall back on asymmetric primitives since it is feasible to reduce their security to some known hard problem. Unfortunately this is not the case with block ciphers such as DES and AES, since their security relies on public scrutiny. It would be interesting to come up with a reduction between the security of white-box implementations and their (black-box) scrutinized security. For example, of great interest would be an obfuscator for which we could say that the white-box implementations of the block cipher is at least as robuust against software attacks, as the original block cipher is against black-box attacks.

It is still an open question if such type of reductions can be achieved.

4.8.3 UWBP for Non-Trivial Families

Let $Q \in \mathbb{PTMF} \cap \mathbb{LF}$. Then it is easy to construct an obfuscator satisfying UWBP for Q with a non-negligible probability (same as that of learning Q). We call such families *trivial*.

Although Theorem 2 rules out the possibility of an obfuscator satisfying UWBP for some $Q \in \mathbb{PTMF} \backslash \mathbb{ALF}$ (which includes most non-trivial families), it does not rule out the possibility of an obfuscator satisfying UWBP for some non-trivial family $Q \in \mathbb{PTMF} \cap \mathbb{ALF}$ (i.e., $Q \in \mathbb{PTMF} \cap \mathbb{ALF} \backslash \mathbb{LF}$). Our next positive result shows that, under reasonable assumptions, this is indeed the case.

UWBP for a Non-Trivial Family

In Theorem 5, we formally capture that there exists an obfuscator satisfying UWBP for a non-trivial (but contrived) family Q.

Theorem 5 *Under reasonable assumptions, there exists a family* $Q \in \mathbb{PTMF} \cap \mathbb{ALF} \backslash \mathbb{LF}$ *and an obfuscator* $\mathcal{O}$ *for* Q *that satisfies UWBP for* Q.

Proof: For simplicity, we prove the above result in the random oracle model. Let $\mathcal{R}_{|q|}$ be a random oracle mapping arbitrary strings to $|q|$-bit strings. Consider the family Q defined using Algorithm 9.

Algorithm 9: Family $Q \in \mathbb{PTMF} \cap \mathbb{ALF} \backslash \mathbb{LF}$.

Function $Q[q]$ *(**Input** Y)* **{**
/* $\mathcal{R}_\ell : \{0,1\}^* \mapsto \{0,1\}^\ell$ is a random oracle */
if $(\mathcal{R}_{|q|}(q||Y) = q)$ **then** output 1 **else** output 0
}

Observe that $Q \in \mathbb{PTMF} \cap \mathbb{ALF} \backslash \mathbb{LF}$. It can be proved that $\forall D \in \mathbb{PPT}$ (the distinguisher),

$$\left| \Pr \left[\begin{array}{c} b \xleftarrow{R} \{0,1\}; q_0, q_1 \xleftarrow{R} \{0,1\}^k \cap \mathcal{K}_Q \\ : D^{Q[q_b]}(1^k, q_0, q_1) = b \end{array} \right] - \frac{1}{2} \right| \leq negl(k), \tag{4.1}$$

the probability taken over the coin tosses of D. For any k, let $q \xleftarrow{R} \{0,1\}^k \cap \mathcal{K}_Q$. Consider an obfuscator $\mathcal{O}$ that takes in as input (Q, q) and simply outputs a description of $Q[q]$ as the obfuscation of $Q[q]$. Let $sn \in \mathbb{SPEC}$ be such that $Q \in_{obf} sn$ but $\mathcal{O}$ does not satisfy WBP for (Q, sn) w.r.t. some adversary $A \in \mathbb{PPT}$ (that is, the white-box advantage of $(\mathcal{O}, Q, sn)$ is non-negligible), then A can be directly converted into a distinguisher D such that Equation (4.1) does not hold, thereby arriving at a contradiction. □

The approach that has been conceived is similar to the positive result by Lynn *et al.* [122]. One can extend the proof to hold in the standard model, based on reasonable computational assumption, similar to the results shown in Canetti [34] and Wee [186].

4.9 The Case of Probabilistic PTMFs

Our negative results apply because the approximate functionality requirement for obfuscators (in the correctness definition of Sect. 4.6.1) is defined only for deterministic PTMFs. What about extended models (such as in [91, 93]) which allow probabilistic families? It turns out that similar results can be obtained for such models using appropriately extended definitions. We give an overview below.

There are two types of PPTMs considering the way randomness is encoded: (1) randomness encoded in the input tape as in the encryption oracle of the example in Algorithm 6, or (2) randomness encoded in the key tape as in the challenge oracle of the same example. Our negative results assume that the obfuscated family is of the first type. However, we can also talk about obfuscating the second type of families (this was considered in [91, 93]). Denote a PTMF

of the latter type a probabilistic PTMF (PPTMF). We ignore any additional randomness in our discussion since the adversary cannot be trusted to supply randomness.

Intuitively, a PPTMF is simply an ordinary PTMF Q with part of the key used for randomness, so that two different keys are "equivalent" provided only their random bits are different. Formally, a PPTMF is a pair (Q, τ), where $Q \in \mathbb{PTMF}$ and τ is an equivalence relation on $\mathcal{K}_Q$ that partitions $\mathcal{K}_Q$ into equivalence classes, s.t. $\forall q_1, q_2 \in \mathcal{K}_Q$:

$$\tau(q_1, q_2) = 1 \iff \text{only random bits of } q_1, q_2 \text{ are different.}$$

In Definition 23, we extend the definitions of equality, (approximate) learnability, approximate functionality and correctness from PTMFs onto PPTMFs using the relation τ.

Definition 23 *In the following, let* $(Q, \tau) \in \mathbb{PPTMF}$.

1. *Let* $q \in \mathcal{K}_Q$, *and* $\mathcal{I}$ *be the input space. Then:*

   - *For any* $(a, z) \in \mathcal{I}_{Q,|q|} \times \{0,1\}^*$, *we say that* z *is* τ**-equal** *to* $Q[q](a)$ *(written* $z =_\tau Q[q](a)$*) if*

   $$\exists q' \in \mathcal{K}_Q : z = Q[q'](a) \wedge \tau(q, q') = 1\,.$$

   - *For any* $X \in \mathbb{TM}$ *we say that* X *is* τ**-equal** *to* $Q[q]$ *(written* $X =_\tau Q[q]$*) if*

   $$\forall a \in \mathcal{I}_{Q,|q|} : X(a) =_\tau Q[q](a).$$

2. *We define a* τ**-(approximate) learnable** *family by replacing "*$=$*" with "*$=_\tau$*" in the definition of (approximate) learnable families. The following claim is easy to prove.*

   Claim 3 *If* Q *is not* τ*-approximate learnable then* $Q \notin \mathbb{ALF}$.

3. *Let* $\mathcal{O} : \mathbb{PTMF} \times \{0,1\}^* \mapsto \mathbb{TM}$ *be a randomized algorithm. Then:*

   - $\mathcal{O}$ *satisfies* τ**-approx. functionality** *for* Q *if*

   $$\forall q \in \mathcal{K}_Q, \forall a \in \mathcal{I}_{Q,|q|} : \Pr[\mathcal{O}(Q, q)(a) \neq_\tau Q[q](a)] \leq negl(|q|)\,,$$

   the probability taken over the coin tosses of $\mathcal{O}$.

   - $\mathcal{O}$ *satisfies* τ**-correctness** *for* Q *if Definition 12 (of Sect. 4.6.1) holds when "approximate functionality" is replaced by "*τ*-approximate functionality".*

- $\mathcal{O}$ *is a* τ**-obfuscator** *for* Q *if it satisfies* τ*-correctness for* Q*.*

4. Q *is* τ**-decidable** *if there exist* $p, p' \in \mathbb{P}$ *and for every* $k \in \mathbb{N}$*, there exists an efficiently computable map*

$$\mathbf{f}_k : \{0,1\}^{p(k)} \mapsto \{0,1\}^k \cap \mathcal{K}_Q \times \mathbb{PTM}\,,$$

such that for all $(q, Z) \leftarrow \mathbf{f}_k(r)$*, the following holds:*

- $\forall a, z \in \{0,1\}^* : Z(a,z) = 1 \iff z =_\tau Q[q](a)$*.*
- *If* r *is uniformly distributed then so is* q*.*
- $|Z| \leq p'(k)$*.*

We claim that for any meaningful PPTMF construction (Q, τ)*, the family* Q *must be* τ*-decidable.*

Despite several known positive white-box results for PPTMFs [91, 93], our negative results also extend to such families due to the 'τ-decidability' property, which roughly says that for every equivalence class of Q in τ, there must exist an efficient distinguisher that decides whether a given PTM is a member of that class or not (any meaningful PPTMF must satisfy this property). Theorem 6 describes the corresponding statement of our main negative result in Theorem 2.

Theorem 6 *For every* $(Q, \tau) \in \mathbb{PPTMF}$ *with* Q τ*-decidable but not* τ*-approx. learnable, there exists* $spec \in \mathbb{SPEC}$ *such that* $Q \in_{obf} spec$ *but every* τ*-obfuscator for* Q *fails to satisfy the WBP for* $(Q, spec)$*.*

The τ-decidability property allows us to extend the counter-example in the proof of Theorem 2.

4.9.1 An Open Question: WBP and Soundness

Let $((Q, \tau), sn) \in \mathbb{PPTMF} \times \mathbb{SN}$ such that the following is true:

1. Q is not τ-approximate learnable.
2. $Q \in_{obf} spec$
3. Q is τ-decidable
4. O is a τ-obfuscator for Q satisfying IND-soundness (Sect. 4.6.2, Definition 13).

A useful question is: *Given a security notion sn, can we decide if $\mathcal{O}$ satisfies WBP for (Q, sn)?*

Why is it useful? Ideally, we would like the WBP to be satisfied. However, WBP is defined with respect to a family and a security notion while soundness is defined with respect to a family, independent of the security notion. On the one hand, due to this simplified definition, obfuscator designers may find it appealing. On the other hand, it is possible that IND-soundness may be too strong to be satisfied even though WBP is (with respect to some security notion), as in the example in the proof of Theorem 4. Nevertheless, we consider it an interesting question to characterize cases when the WBP can be reduced to IND-soundness. The assumption that $\mathcal{O}$ is a τ-obfuscator (rather than an obfuscator) for Q is necessary to falsify Proposition 2, which rules out interesting families.

An Open Question: If *sn* is such that the only oracle available to A is the one being obfuscated, then the WBP for *sn* indeed holds if IND-soundness holds. The result also holds if *sn* has additional oracles that always output the same string (which can be given as an auxiliary input to the distinguisher – cf. "distinguishable attack property" of [93]). At this stage, an open question is: *how to characterize sns where A is given additional oracles which output query-dependent strings?*

4.10 Conclusion

The goal of White-Box Cryptography is to implement cryptographic primitives in such a way that they achieve a certain level of robustness against an adversary that has full access to and control over the implementation of the primitive. Up to some extent, this is related to code obfuscation, which attempts to hide certain characteristics of a program. Despite the fact that many formal models for obfuscation have been presented, white-box cryptography lacks foundations. This chapter provides an initial step to bring the foundations of white-box cryptography to a same level as obfuscation.

Our work made several contributions in this regard. We extended the notion of WBC to arbitrary cryptographic primitives and initiated a formal study of WBC by introducing precise definitions of what it means for a white-box implementation to be secure. To achieve this, we formalized the White-Box Property (WBP). The WBP is defined for a program family (e.g., encryption) with associated *security notion* (e.g., IND-CPA), and describes how much 'useful information' is leaked from the white-boxed program. It captures the security requirements of WBC defined over some scheme and a security notion. We also showed how to encode security notions in a formal manner, which might be of independent interest.

This new theoretical model provides a context to investigate the security of

white-box implementations. We present some (im)possibility results, and describe the connection between WBC and obfuscation. We have showed that WBP and soundness are in general quite independent of each other by giving some examples where one is satisfied and the other is not. Although the WBP is defined for a particular (family, security notion)-pair, soundness is only defined for a given family and is independent of the security notion. A natural question is whether there exist non-trivial families for which the WBP with respect to every security notion can be reduced to the soundness of an obfuscator for that family. Loosely speaking, an obfuscator that achieves this is said to satisfy the *Universal White-Box Property (UWBP)* for that family. We showed that the UWBP fails for every family that is not approximate learnable, in the sense that there exists a (contrived) security notion that is satisfied in a 'black-box' setting, but fails when an adversary has white-box access to the obfuscated program. However, we show that under reasonable assumptions there exists an obfuscator $\mathcal{O}$ satisfying UWBP for a non-learnable but approximate learnable family.

Since the security notion used for our negative result is quite contrived, it might seem reasonable to expect that a meaningful notion of security for WBC based on WBP can still be achieved for 'normal' security notions. We have show that under reasonable computational assumptions, there exists a non-learnable family and an obfuscator that satisfies the WBP for that family under a meaningful security notion. In particular, we described an obfuscator that turns a IND-CPA secure symmetric scheme into an IND-CPA secure asymmetric encryption scheme, hence proving that white-box cryptography does exist under reasonable assumptions for reasonable families of functions and security notions.

Chapter 5

Applications

Cryptology is a tool to secure our information infrastructure. It is of importance that academic results in the research of cryptology (even if they might be rather theoretical), are conceived within a broad practical context. In this chapter we wish to present a selection of applications and techniques, that can benefit from and contribute to the research in white-box cryptography.

For some of the applications, we assume that *reasonably secure* white-box implementations exist. That is, for the application that is elaborated upon, the white-box implementation is sufficiently resistant against adversaries that target that particular application. The discussion on feasibility and constructions are the subjects of Chapter 3 and 4 respectively, and are taken for granted here.

Overview. There is a vast range of applications that can benefit from research in white-box cryptography. This is natural due to the research problem that white-box cryptography aims to address. In this chapter, we elaborate on a selected set of applications, grouped into four different research domains that we distinguish:

1. New and improved cryptographic primitives – The primary objective of white-box cryptography is to hide key information. Hence it has a natural behavior to convert symmetric primitives into asymmetric primitives.

2. Involvement of hardware – In many applications, hardware is deployed as a secure device (e.g., for TV set-top boxes). Using software protection techniques such as white-box cryptography, some tasks can be shifted towards software implementations. On the other hand, hardware can be used to address fundamental issues in a pure software model, in particular the

problem of cloneability. Secondly, the techniques used to protect hardware implementations can benefit from white-box techniques, and vice versa.

3. Computing in the encrypted domain – We show that there exists a relation between white-box cryptography and techniques that are related to computations on encrypted data. This includes homomorphic encryption schemes and techniques for secure function evaluation.

4. Software protection – The objective of white-box cryptography is related to code obfuscation research: both aim to protect information in software implementations. Although we wish to distinguish between the two concepts, we show how ideas from white-box cryptography can be leveraged to software protection techniques including diversification, fingerprinting, tamper resistance, and trustworthy execution of software.

We elaborate on each of these domains in the corresponding sections below, and present some practical applications for each class. Remark that some applications can benefit from white-box cryptography from different directions. In the final section, we present the *Digital Rights Management (DRM)* class, which is a controversial application class of specific interest to white-box cryptography, that benefits from WBC in different ways.

5.1 New and Improved Cryptographic Primitives

5.1.1 Asymmetric Encryption Schemes

One particular use that we have mentioned before, is the ability of white-box cryptography to convert symmetric primitives into asymmetric primitives.

In 1986, Desmedt and Quisquater [56] suggested to simulate public key encryption schemes in closed systems (e.g., smart cards). In particular, they propose to derive public key cryptography from symmetric schemes in a tamperproof hardware model. This is hardware which has the property that one cannot get the secret out.

When public-key cryptography was first proposed by Diffie and Hellman [59], they suggested that one way to produce a public-key encryption scheme was to obfuscate a secret-key scheme. This application of obfuscation was also suggested by Barak *et al.* [6], while the transformation of a MAC into a signature scheme was also suggested by Joye [102], and formally discussed by Hofheinz *et al.* [91], Hohenberger *et al.* [93], and Saxena and Wyseur [165]. Concrete examples were provided how to turn a symmetric cipher (E, D) into an asymmetric cipher, by obfuscating the instantiated encryption algorithm E_k. As a result, all parties that are in possession of the public key $\mathcal{O}(E_k)$ are able to encrypt, while only

the party in possession of the private key k is able to decrypt. Similarly, a MAC algorithm can be converted into a signature scheme.

Note that, in order to deploy such a transformation, the extended Definition 3 (see Sect. 3.1.3) needs to be adopted, and security notions such as IND-CPA and IND-CCA need to be satisfied. In the proof of Theorem 4, we presented a positive result on the obfuscation of a symmetric primitive that satisfies IND-CPA and results into an asymmetric primitive that satisfies IND-CPA. However, the example was obtained with building blocks that were asymmetric in the first place. It remains an open question whether constructions can be proposed that consist of building blocks that have not been used in asymmetric cryptosystems before. Hence, it remains an open question if white-box cryptographic can be used as a generic tool for generating innovative asymmetric cryptosystems. On the other hand, white-box implementations might offer a great deal of flexibility. For example, a white-boxed public key encryption scheme could be tweaked as such that one part (e.g., the private decryption functionality) is small and efficient (which could meet constraints of lightweight hardware devices), compensated by a large counterpart for encryption.

5.1.2 Programmable Random Oracles

The *Random Oracle Model* (see Bellare and Rogaway [10]) is an idealized cryptographic setting in which all parties (including the adversary) interact with each other, and in addition can make oracle queries. All queries to the random oracle, regardless of the identity of the party making them, are answered by a single function that is selected uniformly at random from all possible functions. The set of possible functions $f_s : \{0,1\}^* \rightarrow \{0,1\}^{l\text{out}^{(s)}}$ is determined by the output length function l_{out} and by the security parameter s. A more stringent notion of a random oracle, is where the inputs are of a pre-determined (short) length. That is, functions of the form $f_s : \{0,1\}^{l\text{in}^{(s)}} \rightarrow \{0,1\}^{l\text{out}^{(s)}}$.

This is a very controversial model, since real-world systems do not have access to such random oracles (there is an infinite amount of possible functions). Hence, there is a need to implement the random oracle of the idealized setting. That is, replace the random oracle by a easy-to-evaluate function that is directly available to each party. It is hoped that cryptographic primitives that are designed in the random oracle model, remain secure when the random oracle is replaced by this *efficient, public* function. Denote a random oracle that can be implemented in real-world systems as a *programmable random oracle*.

While the Random Oracle methodology is an abstraction of cryptographic applications that offers great advantages towards security analysis, a great deal of caution is needed when implementing the application in practice. Unfortunately, it has been shown that schemes that are secure in the random oracle model have

insecure instantiations (e.g., by Canetti *et al.* [36]), while it remains unknown if there exist programmable random oracles with strong enough properties for specific primitives.

It has been proposed to implement random oracles by obfuscating a family of pseudorandom functions (MAC algorithms or digital signature algorithms for arbitrary input; encryption schemes for the stringent notion), whose input-output behavior is by assumption computationally indistinguishable from that of a truly random function. The pseurorandomness of the output in black-box is transcended to the obfuscated implementation due to the functionality equivalence requirement in Definition 5. Although the RO model does not by itself provide any security guarantees for implementations in the standard model, one might hope that the obfuscation of pseudorandom functions, leverages a notion of security from the RO model to a notion in the standard model for a large class of natural cryptographic applications.

The availability of programmable random oracles opens directions towards practical access control systems (see Lynn *et al.* [122]), password managers (see Wee [186]), and digital vaults (see Canetti [35]), because the programmable random oracle can be used for secure point function obfuscation.

5.2 Hardware

A natural answer to the issue of hiding key information would be to deploy hardware. Hardware devices can store sensitive data, and/or perform computations in a location which is shielded from the adversary. Examples of hardware devices that can be deployed are USB dongles, Smart Cards, Trusted Platform Modules (TPM), and cryptographic co-processors. In Table 5.1, we present a brief overview of some of the advantages and disadvantages of deploying hardware in practical applications.

Hardware-based solutions may be vulnerable to side-channel analysis (see Sect. 2.3) and imply a cost in terms of flexibility, deployment, revocation and replacement. We wish to address these issues by deploying software-only implementations, and protect them with obfuscation and white-box techniques. The cost of software solutions is very small, since they can be transported electronically. They also carry none of the administrative and physical limitations of hardware solutions. However, software might have some disadvantages as well. In a full access model (such as the white-box attack context), software can be cloned and tampered with easily. Moreover, hardware is much more suitable to address entity authentication issues.

Our main objective is to consider the security of cryptographic primitives in

Table 5.1. Comparison between hardware and software

Pro hardware	Contra hardware
"Safe" heaven Tamper resistant (up to some level) Enables authentication • users (Smart Cards) • platforms (Trusted Platform Modules) Hard to clone.	Lack of flexibility Cost Possible malicious behavior Only a partial solution, or even inapplicable Side-channel analysis

a software-only situation. Nevertheless, both fields of *secure hardware implementations* and *white-box cryptography* can benefit from each other.

Hardware-based solutions to enforce software protection techniques. On the one hand, solutions that are developed for secure implementation of algorithms (circuits) in hardware, can be inspiring for white-box cryptography. Observe that the random bijections that are introduced to obfuscate the lookup tables of the white-box DES and white-box AES implementations, can be seen as an extension of masking techniques [131], developed to prevent side-channel analysis.

Secondly, we might consider to re-introduce (lightweight) hardware to address software-related issues. A particular motivation for the re-introduction is to enable authentication, and to prevent code cloning. For example in [77], Goldwasser *et al.* proposed to use hardware, to introduce a constraint on the number of uses of a (obfuscated) program. This is in particular useful in the case of DRM, where a party could be allowed only up to t executions of an implementation. Securely obfuscating some count-down functionality obviously contradicts with the adversary's capability to clone the functionality.

White-Box Cryptography to enforce hardware implementation protection techniques. Conversely, white-box cryptography might also aid in the development of protection techniques against side channel attacks. As mentioned in Sect. 3.1, the white-box attack context is the worst-case attack model, and therefore is a superset of any side-channel attack model. Hence, white-box implementations wired into hardware devices are by definition resistant to *any* side-channel analysis. Unfortunately, the footprint and efficiency in hardware of current white-box implementations is dramatic. Weakened white-box implemen-

tations can be considered, such as implementations where only (critical) parts are obfuscated, and white-box implementations of ciphers that have a smaller footprint when obfuscated.

For example, a side-channel resistant vault on a smart card can be constructed, by implementing a (small) block cipher on the smart card, and storing the lookup tables in the smart card's EEPROM. Similarly, cryptographic operations on mobile devices (such as mobile phones and PDAs) and TV set-top boxes can be protected. However, one should be very careful with such an approach, since an adversary could deploy a strategy in two phases: (1) extract the lookup tables from the EEPROM by side-channel analysis, and then (2) deploy a fully privileged attack on the reconstructed implementation.

From a research point of view, it is of interest to investigate how design proposals and analysis techniques for white-box implementations carry over to other research areas in cryptology, such as side-channel cryptanalysis. For example, a comparison between white-box AES implementations and protection techniques against cache attacks has been identified in [12].

5.3 Computing in the Encrypted Domain

5.3.1 Homomorphic Encryption

Homomorphic encryption schemes refer to a special class of (probabilistic) asymmetric encryption schemes, where algebraic operations on plaintexts can be performed directly on the respective ciphertexts. That is, for some algebraic operation $+$ on plaintexts $p_0, p_1 \in \mathcal{P}$, one can perform an algebraic operation $\otimes$ on $\mathcal{E}(p_0), \mathcal{E}(p_1)$, such that

$$D(\mathcal{E}(p_0) \oplus \mathcal{E}(p_1)) = p_0 + p_1 \,.$$

Goldwasser and Micali [78] introduced a first homomorphic encryption scheme, based on the Quadratic Residuosity Problem, where the modular addition of two bits can be performed in the encrypted domain by a multiplication of the corresponding ciphertexts. Several improvements were presented to address performance issues, and increase the bandwidth ratio.[1] The most notable scheme is the Paillier cryptosystem [144], on which many cryptographic schemes for applications related to *privacy-preserving cryptography* are constructed. In particular, Ostrovsky and Skeith [143] introduced a construction which implements a private information retrieval system that is in fact an obfuscation of a filtering system.

[1] The bandwith ratio can be defined as the amount of information that is embedded in a ciphertext.

A long standing open problem is whether *ring/algebraic homomorphisms* exists. That is, an encryption scheme that satisfies the homomorphic property over two operations $+, \times$:

$$\forall p_0, p_1 \in \mathcal{P} : D(\mathcal{E}(p_0) \oplus \mathcal{E}(p_1)) = p_0 + p_1 \,,$$
$$\text{and } D(\mathcal{E}(p_0) \otimes \mathcal{E}(p_1)) = p_0 \times p_1 \,.$$

No convincing algebraic homomorphic encryption scheme has been found yet. One of the best results so far, is a scheme by Boneh *et al.* [26], where additions can be (almost) freely performed in the encrypted domain, and one multiplication. As such, polynomials of degree 2 (and hence circuits of depth 2) can be computed in the encrypted domain. The construction by Boneh *et al.* is based on *pairings*, and it seems hard to extend this to higher degree computations. Recently, Gentry [69] presented a homomorphic encryption for circuits, based on Lattice theory [155]. It remains to be seen up to what extent this is a step towards a ring homomorphism.

White-box cryptography could be used in practice to obtain a *weak* asymmetric encryption scheme that can be used as an algebraic homomorphism. Weak in the sense that we should not hope this results in strong IND-CCA2 secure schemes. This could be achieved by white-boxing some special re-encryption scheme such that

$$Hom(c_0, c_1, r) := (\mathcal{E}_K(D_k(c_0) \times D_k(c_1), r)) \,,$$

where k is the private decryption key that needs to be protected, and the Hom operation denotes the operation in the encrypted domain, corresponding to $\times$. We acknowledge that this approach is subject to a lot of criticism. However, its deployment within some larger application can be beneficial to raise the bar against adversaries.

5.3.2 Secure Function Evaluation

In Sect. 3.2.2, the concept of *Secure Function Evaluation* was introduced as a related concept. Its objective is to compute some given function f, on the inputs (x_A, x_B) of two parties A and B, where both parties want to keep their inputs secret. To achieve this, one party (Alice) creates a *garbled circuit*, and sends it to the other party (Bob). A garbled circuit is constructed as follows: consider a circuit that computes the function f, and assign random (garbled) values to each value of every wire. Then, construct *garbled tables* that compute some gate, given garbled input values, and computes a garbled value of the corresponding output. Bob is able to evaluate this garbled circuit with his private inputs and compute the result. This process of evaluation requires that Bob receives the (garbled)

values of Alice, without revealing any private information to each other. This is achieved via *Oblivious Transfer (OT)* [152]. For the initial constructions [73], OT operations were required for every evaluation of a $\wedge$ gate ($\vee$ gates could be performed via a homomorphic operation). Moreover, these constructions were designed within the context of an *honest-but-curious adversary*. Subsequent work addressed these issues by introducing a *cut-and-paste* technique to enforce honest behavior, and reduce the number of communication rounds [164, 111, 117].

White-box cryptography aims to achieve something similar; namely, to generate an implementation that is randomized (garbled), contains some confidential input k (x_A), and can be evaluated securely by another party on its input p (x_B) with a minimal of communication (preferably none, besides the delivery of the obfuscated program). Similar to the *universal circuit* trick by Valiant [182], we wish to construct the garbled circuit for $f(k, \cdot) = E_k(\cdot)$.

Unfortunately, such circuits seem to be difficult to obtain for block ciphers. The fundamental problem is that garbled circuits should be able to evaluate any input (x_1, x_2). Hence, within the context of encryption functions E_k, such circuits should be able to encrypt plaintexts p for any k. In the case of a garbled white-box AES implementation, all garbled S-boxes alone would already occupy $10 \cdot 2^{95}$ bit $= 10 \cdot 2^{62}$ GByte of memory size, far beyond our reach. Moreover, white-box cryptography demands standalone solutions. Nevertheless, *Secure Function Evaluation* remains a very interesting topic of study, where further collaboration with research on white-box cryptography might result in innovative results.

5.4 Software Protection

White-box cryptography can be seen as a particular instance of code obfuscation. Where in code obfuscation, the goal is to hide the make it hard for an adversary to understand the code (mainly against reverse engineering), white-box cryptography aims in the first place to hide secret keys in software implementations of cryptographic algorithms.

The techniques that have been developed to achieve this goal can be deployed in a context beyond cryptographic primitives, and can be beneficial for a large range of software protection techniques. Obviously, they can be deployed to improve the protection of data values, and to randomize the internal data flow for a generic class of programs. Furthermore, white-box techniques can contribute to software protection techniques such as *software tamper resistance*, *software diversity*, *data flow obfuscation*, and so forth. For an overview of protection techniques, we refer to our survey in [38]. Below, we present some software protection techniques, and introduce how they can benefit from white-box cryptography techniques.

5.4.1 Software Tamper Resistance

One of the major problems in software protection is to develop tamper resistant software (TRS). The objective is to protect the assets of a software implementation against illegitimate use. Assets include the program code, partial functionality (e.g., an internal decryption routine), internal variables, and cryptographic keys. In [137], Nagra, Wyseur, and Herlea present a formal study on the assets and goals of such an adversary. In order to defend these assets, typically, code obfuscation techniques are deployed [179].

However, in contrast to code obfuscation techniques, which prevent against reverse engineering, tamper resistant techniques need to prevent against targeted modifications that aim to eliminate functionality of a program (e.g., disable some verification routine), or change operating parameters. Obfuscation only supports TRS, in making it harder for an adversary to find out where exactly to tamper program code.

Software self-checking. Two main families of tamper resistance techniques can be identified. A first family includes techniques that introduce verification techniques (based on hashing). Software self-checking techniques introduce software guards that frequently verify the integrity of the code. Horne *et al.* [95] suggested to introduce a complex network of small software guards that (besides the code) also verify each other's integrity, hence making it hard to remove the protection technique.

Unfortunately, all self-checking suffers from the cloning attack [183]. Observe that at execution of a program, code is *executed*, while in a verification process, code is *read*. In the cloning attack, the original program is cloned before its code is tampered with, and the code read processes are redirected to the (untampered) cloned program, hence the tampered program can execute without tamper detection. This can be achieved easily on Von Neumann architecture machines (which most modern computers are), which has a natural distinction between code-read(/write) and code-execute. One countermeasure, presented by Giffin *et al.* [70], is to strengthen software self-checking by introducing self-modifying code. This issue can be solved in a networked scenario, within the context of *trustworthy execution* (see Sect. 5.4.3).

Tamper resistance by software hardening. A second family of tamper resistance techniques does not include verification, but aims to make sure that upon tampering, the execution results are useless. This family is in particular interesting when a cryptographic primitive is included in the application. Consider a DRM client, where encrypted content is processed by a decryption routine after authentication. In such an application, the decryption routine would return

random junk instead of valuable content upon tampering (of the authentication code).

In [134], Michiels and Gorissen presented *Medusa*; this technique takes advantage of the redundancy introduced by the annihilating encodings in white-box implementations, to achieve some degree of software tamper resistance. The main idea is to make a binary program code tamper resistant by incorporating the code into the descriptions of the lookup tables of a white-box implementation. As a result, the code of a protected application has a dual interpretation: it is both program code as decryption key. Hence, tampering of the code results in a modification of the decryption key. Note that techniques like this only verify the integrity when a encryption/decryption is performed, which is in particular interesting for DRM applications. Once the final encryption/decryption has been performed, this technique does not add any protection to the remaining code. Other limitations are that an adversary can attempt to repair the key, by introducing new annihilating encodings, or that an adversary tries to break the dual interpretation by tampering addresses to point to a cloned (untampered) white-box decryption routine, similar to the strategy described in [183]. Nevertheless, this remains an interesting study subject.

5.4.2 Diversity

In the battle against malware, an important tool is diversification of software implementations. Diversification produces different (orthogonal) instances with a similar functionality, such that a *crack* on one instance, cannot be mounted on another instance. The injection of annihilating encodings introduced diversity of white-box implementations, and might bring diversification techniques to other natural classes of programs. For example, consider a DRM client introduced above, where the decryption routine is white-boxed and includes an external input encoding that is annihilated in the preceding authentication code. If different instances of the DRM client are compiled by choosing different external input encodings, the client becomes resistant to cracks that attempt to disable the authentication code (up to some extent). Further research is required to explore the full capabilities of white-box cryptography for software diversity.

5.4.3 Trustworthy Execution

One of the principle challenges in software protection, is the problem of *trustworthy* execution of software on an untrusted platform, where a verification entity is able to assure correct execution of software. We define the problem of *remote code integrity verification* as the act of delivering attestations (proofs) to a verification entity that guarantee code executes untampered on a remote entrusting platform. On such a platform, an adversary has administrative privileges and can

tamper with all the software including the operating system; this is the white-box attack model.

So far, establishing a trusted execution environment on an untrusted platform has been an open research challenge. An adversary having complete control over an untrusted platform also has control over its input and output traffic. This makes it difficult for a verifier to be assured of communicating with a particular environment on an untrusted platform. Even more: to be guaranteed software is actually running in that environment.

Trusted Computing

A first approach is to introduce hardware tailored specifically to provide assurance on an untrusted platform. The Trusted Computing Group (TCG) defines the addition of a Trusted Platform Module (TPM). A trusted computing platform reliably measures the software (i.e., BIOS, bootloader, operating system components and user applications) that get loaded during startup of the platform [160], and can later report its configuration to a remote entity with an attestation protocol.

The TCG attestation process has deployment issues (see Sadeghi and Stüble [159]) and requires a secure operating system. Virtualization technology like Intel Trusted Execution Technology (TXT) [83] and Hypervisor [49] can overcome some of these shortcomings. Virtualization is an emerging trend, supported by the increased availability of communication networks and open computing platforms.

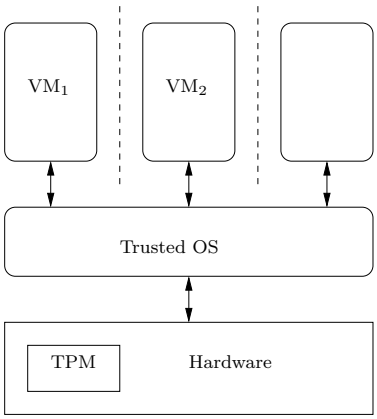

Figure 5.1. The Hypervisor Architecture

Fig. 5.1 depicts the basic Hypervisor architecture, where several virtual machines (VM) can run on a single platform. The core of the Hypervisor architecture is a trusted operating system, enforced by a TPM, which makes sure the virtual machines remain isolated and can perform their operations as specified. This (mini) operating system does not have any root access, and is obfuscated. To guarantee trustworthy execution of the trusted operating system, attestation techniques are deployed (which require network access). An interesting direction, which is subject of current research on Hypervisor systems, is the 'free' migration of virtual machines from one computing platform to another. Unfortunately, this remains a difficult task to achieve, because this also implies that secret keys (which are bound to the TPM of the computing platform) need to be able to migrate also. Therefore, an interesting subject of further research is to investigate how white-box cryptography can aid in the protection of secret keys for the trusted OS, by shifting some tasks from the TPM to a software level, and aid in the migration of virtual machines.

Another direction is to use white-box cryptography as a fall-back for a TPM [158]. The main idea is that when a TPM breaks down, the content that was encrypted with the TPM can still be decrypted by means of the white-box implementation, which has embedded the cryptographic key of the TPM (that was instantiated when the TPM was instantiated). To make sure an adversary cannot call the white-boxed decryption oracle, authentication needs to be added. In the current state of the art, when a TPM breaks down, the manufacturer needs to be requested for the decryption key, which could involve a difficult procedure. Also the manufacturer would prefer not to be bothered with this, or could ask for a service fee.

Remote Entrusting

In the RE-TRUST project [154], the problem of trustworthy execution of software is addressed in a pure-software scenario. The context is a network scenario, where a continuous network connection between a trusted server and the untrusted platform is available. The objective is to enable the trusted server to verify (at execution time) whether or not software running on a remote platform is executing *as specified*. If malicious behavior is detected (e.g., an adversary is tampering with the software), appropriate actions can take place (e.g., the trusted server can shut down its service, or force the client software to stop its execution). This paradigm was coined *remote entrusting*.

White-box cryptography is a cornerstone in this remote entrusting paradigm, because cryptographic primitives (accompanied by secret keys) need to be deployed in the client software implementation, and attestations need to be signed.

Other techniques that are deployed include dynamic updating, to make sure that the software cannot execute without the network connection. Hence, the network connection can be seen as a sort of 'lifeline'.

The remote entrusting paradigma is of growing importance, given the increased availability of communication networks and a trend towards open computing platforms. Since standalone solutions often do not suffice to achieve the high security requiments of new software applications, it is of great value to develop networked solutions. Target applications include health care applications, banking applications, and on-line games. In practice, a trusted entity would monitor the execution of client applications to verify its trustworthyness. Such technology is currently deployed in the gaming industry, e.g., in the popular game World of Warcraft (WoW), where a game server would only accept game clients to connect, when these clients are trustworthy. We refer to McGraw and Hoglund [128, 92] for an extensive study on the threats on online games (applied to WoW), and software security technologies that are deployed to protect them.

Remote attestation on legacy operating systems with trusted platform modules. Pure software-based solutions have been proposed to address the cloning attack in a remote verification scenario. Examples are Pioneer [170] and TEAS [68], which incorporate time measurements in the verification process. Unfortunately, these suffer from network delay. On the other hand, TCG attestation techniques require the deployment of TPMs and custom operating systems. In [166], Schellekens, Wyseur, and Preneel proposed a remote attestation technique for legacy operating systems that aim to address the disadvantages of pure software-based solutions and TCG-based solutions. Our assumption is the availability of a TPM, which we claim is a reasonable assumption because new modern computing platforms are usually equipped with a TPM.

In Fig. 5.2, we present a time overview of our TPM-based remote attestation technique. The main idea is to use the TPM clock ticker to time the duration of the software verification. When the adversary (A) performs other operations during the verification process (such as redirecting read operations to an untampered version), the verifier (V) should be able to detect this. To avoid pre-computation or replay, the adversary challenges the verification process by sending a random nonce n. The nonce is used by the TPM to create a timestamp *TS1*, which is used as a seed for the self-checksumming operation (`chksum()`). The result c is timestamped again by the TPM (*TS2*), and sent to the verifier, who will *trust* the result only when

$$t_2 - t_1 < \Delta t_{\text{expected}} = \Delta t_{\text{chksum}} + \Delta t_{\text{sign}} + \delta t\,,$$

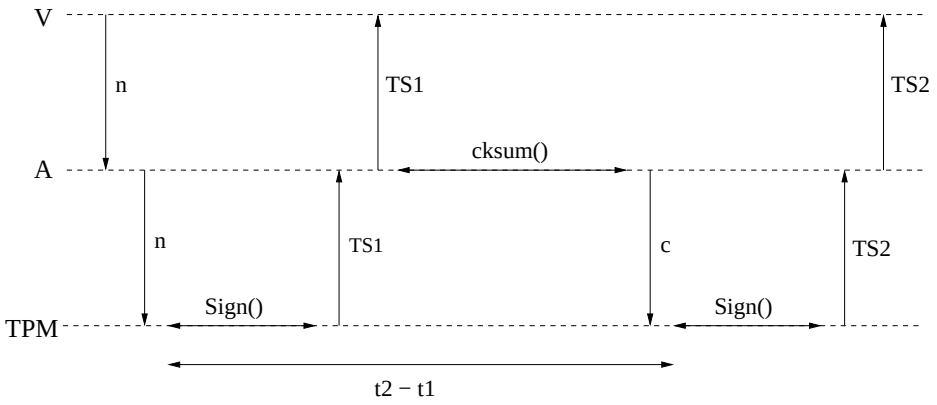

Figure 5.2. Time Overview of the Improved Pioneer Protocol

where Δt_{chksum} depends on the specifications of the execution platform. To avoid hardware upgrades (e.g., a faster computation of `chksum` might give the adversary sufficient time to introduce redirections of read instructions), a benchmark verification can be computed at boot time (initiated by a trusted bootloader). We refer to [166] for technical details and further improvements.

White-Box Remote Program Execution (WBRPE)

In [89] Herzberg *et al.* present a new model for secure remote execution of software, where the the remote execution platform is subject to white-box attacks. The research on *white-box remote program execution (WBRPE)* was motivated and inspired by the theoretical impossibility results on obfuscation. The results on WBRPE are rather theoretical as well at this point, but nevertheless intended for practical deployment. The main idea is to circumvent the negative results on obfuscation [6] by relaxing the functionality requirement.

Roughly speaking, the architecture (depicted in Fig. 5.3) is specified as follows: on the untrusted host, an *Obfuscated Virtual Machine (OVM)* is deployed. The OVM is able to process encrypted input from the trusted server (the specification of a program P, optionally with some additional confidential input), and an input a from the untrusted host. The OVM outputs the encrypted result of the evaluation of P with the corresponding inputs. We refer to Herzberg *et al.* [89] for a more detailed specification.

The adversary has white-box access on the untrusted host, that is, full control over the OVM (and its execution platform), the encrypted input $E(P)$ and the

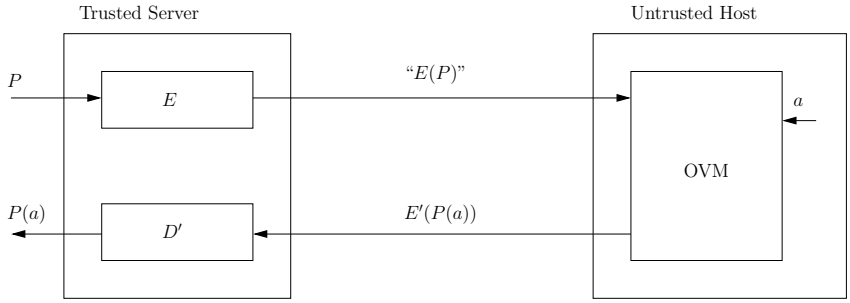

Figure 5.3. White-Box Remote Program Execution Architecture

encrypted result $E'(P(a))$. The aim of the architecture is to achieve *trustworthy execution* on the untrusted host, that is, to achieve confidentiality on the program P (the objective of obfuscation) and the result $P(a)$, and to detect tampering. Confidentiality of P is achieved by means of encrypting P, and the assumption of a secure implementation of the virtual machine (therefore denoted as obfuscated virtual machine). Tamper resistance is obtained by the authenticity of the result $P(a)$, achieved by encryption of the result using an encryption scheme that provides data authentication (e.g., encrypt + MAC). Authenticated encryption schemes are encryption schemes that simultaneously protect the confidentiality and authenticity (integrity) of data. When tampering occurs (modification of $E(P)$, tampering of the OVM, or modification of $E'(P(a))$), the trusted server should be able to detect this when verifying the authenticity of the result (up to some negligible probability).

In other words, the goal is to capture that WBRPE is as secure as the underlying encryption schemes used, provided that the obfuscation (of the OVM) is secure. The standard notion of confidentiality is defined using indistinguishability. A similar notion is used in WBRPE, while a definition of integrity is used to capture tamper resistance. Other definitions that are introduced include "privacy of remote inputs", "validity of programs before execution". We refer to [89] for further details.

Because the functionality of the 'obfuscated' program is modified to encrypted output instead, the impossibility results of Barak *et al.* do not apply. Hence, one can hope that any program can be executed securely on the untrusted host, based on this architecture. A positive result towards *generic* 'obfuscation' within this model, is the theory on *robust combiners for white-box security* by Herzberg and Shulman [88]. The main idea is to consider the OVM, instantiated with an input $E(P_1)$, as a program P_2, and parse it with another OVM.

Obfuscated Virtual Machine

The most important step in the practical realization of the architecture as depicted in Fig. 5.3, is the development of an Obfuscated Virtual Machine, that meets the security requirements of confidentiality (of the program and result) and integrity (of the result). All the other components are executed on a trusted server, and can be constructed using existing technology such as block ciphers and digital signature schemes. One of the motivations for the theoretical model described in Chapter 4, is to investigate the feasibility to achieve both security requirements (captured by security notions) at the same time for some suitable schemes.

The OVM needs to contain a set of secret keys for decryption of P, and authenticated encryption of the result, process the input (decrypt), and evaluate the obtained program P with a given input a without leaking the behavior of the program. The latter is the main challenge, because operations performed by the OVM are directly visible to the adversary.

Moreover, the aim is to enable the OVM to evaluate *any* given program, and hence faces similar problems as *secure function evaluation*. Namely, that obfuscated (garbled) circuits tend to be too large when they need to be able to compute programs of significant complexity and size. Moreover, the requirement to perform an authenticated encryption at the output enforces even more constraints towards implementation size and performance.

A generic approach would be to construct some hypothetical *obfuscated* Turing Machine, that is able to perform secure computations, or circuits that contain *obfuscated gates* as a basic building block. These are custom gates, that comprise of two or more standard gates (e.g., $\wedge$, $\vee$, $\bigoplus$). Besides the input to these standard gates, the new gate should also accept some input that selects which standard gate needs to be evaluated. For example, a $\wedge, \vee$ gate is defined by the truth table in Table 5.2, and we denote such a gate as $\mathrm{SG}^{\{\wedge,\vee\}}$.

Observe that b_2 denotes which gate needs to be evaluated, and b_0, b_1 are the input to that respective gate.

Obfuscating such a *special* gate could be done as follows (inspired by white-box cryptography): choose a random permutation π_{in} over $\mathrm{GF}(2^3)$, and a permutation π_{out} over $\mathrm{GF}(2)$. Then, we denote an *obfuscated* gate as

$$\mathcal{O}(\mathrm{SG}^{\{\wedge,\vee\}}) := \pi_{out} \circ \mathrm{SG}^{\{\wedge,\vee\}} \circ \pi_{in} \,.$$

We claim that an obfuscated gate is *locally secure* when the entropy of the output bits is $\frac{1}{2}$. With local security, we capture that an adversary is unable to distinguish which gate is evaluated (hence, an adversary cannot obtain knowledge of b_2), and that the input bits b_0, b_1 remain hidden (garbled).

A major research question is the composition of such gates. Similar to Barak *et al*'s result, the impossibility of *function composition* presented by Lynn *et*

Table 5.2. The $\mathrm{SG}^{\{\wedge,\vee\}}$ gate

b_0	b_1	b_2	*result*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

al. [122, Theorem 5] does not carry to the composition of obfuscated gates due to the modification in functionality caused by the random permutations. For specific classes of programs (with limited depth), we are able to present a secure implementation that can be extended towards an implementation of an OVM.

An extensive study and comparison with secure function evaluation (using garbled circuits) is subject to future work.

5.5 Digital Rights Management

One of the most controversial, but nevertheless one of the most attractive application domains, is the area of *Digital Rights Management (DRM)*. Digital Rights Management refers to technology that aims to enforce conditional access to, and restricted usage of data (content), programs, and devices. In this section, we show how the different research domains described above can be applied this particular class of applications.

The DRM threat model is similar to the white-box model, in the sense that the usage restrictions need to take place on the adversary's platform. The data, programs, and devices that need to be protected are under control of the adversary. In order to disable direct access, the data, programs, and devices need to be protected. This is achieved by encrypting content, and deploying tamper resistant devices. To enforce the conditional access and restricted usage, a wide range of techniques are typically needed, including access control systems, content protection schemes, trustworthy execution, secure storage, and tamper resistant software.

Popular DRM schemes include Apple's FairPlay system, which is used by the online iTunes store [97], and Microsoft's Windows Media DRM (WMDRM) [136]. These schemes use cryptographic schemes in their implementation to protect the

multimedia content against illegitimate use.

A typical DRM scenario addresses conditional access to multimedia content, where (encrypted) content is publicly available (e.g., in a broadcasting setting), while only a set of legitimate users should be able to access the content. Access to content is achieved by means of decryption, while the decryption routine and secret keys are embedded in the client DRM application (executing on a hostile platform), or hardware tokens (e.g., smart cards). The decryption key must be secret and inaccessible to the adversary. Recovery of the decryption key allows decryption of the content without any restriction, hence defeating the DRM protection (see Sect. 1.2.2). This is a major problem, and many so-called secure DRM schemes have failed to live up to their expectations because they neglect to keep the keys confidential. Examples of schemes that have been breached include the CSS copy protection schemes for DVDs, and the AACS content protection scheme that is used for the HDDVD format [66].

A second threat is the isolation of the decryption routine (code lifting). Instead of extracting the decryption key, an adversary can use the isolated decryption routine to decrypt content, circumventing any authentication and rights verification. Here too, white-box cryptography is a candidate technology, where external encodings (see Sect. 3.2.3) can be used to interweave a decryption routine with authentication code. We can formalize the objective of WBC within the context of DRM as follows:

The objective of white-box cryptography is the guarantee that an adversary has no other option but to 'execute' the software implementation in order to access content, hence enforcing authentication verification and conditional access enforcement.

This objective is related to our Definition 3 (see Sect. 3.1.3), in the sense that we state that white-box cryptography aims to guarantee that an adversary has no other option but to resort to black-box attacks.

In [122], Lynn *et al.* presented access control systems as an application of their positive result on obfuscation, while Goldwasser and Kalai [76] presented obfuscation within the context of controlled delegation of authority and access. The use of white-box cryptography to enable DRM was suggested already from the very start [43], and it has been presented as a key technology for the design of DRM architectures in [193, 133, 185]. In the mean time, white-box cryptography has been commercially deployed to secure DRM applications [175] and TV set-top boxes [44].

5.5.1 Traitor Tracing

Besides enforcing conditional access, an interesting direction in DRM is to discourage users to behave maliciously. This involves the topic of *traitor tracing* [28], where a decryption implementation is augmented by some fingerprint such that the implementation can be traced back to the user to whom it was originally provided. Consider the above described DRM scenario. A major threat is where legitimate users present their decryption keys, or decryption software (along with authentication codes) to unauthorized users, thus giving them the ability to decrypt the protected content. This is by far a bigger threat than re-distribution of decrypted content, in particular for short-lived content (e.g., hyped TV productions).

Traditionally, traitor tracing schemes are designed such that a large set of *user decryption keys* can be derived from a master key, where every user key can be used to decrypt the same content (when some extra conditions are met). A traitor tracing scheme is called t-secure when an adversary cannot create a new decryption key when given up to t existing user decryption keys, without revealing one of the keys it is derived from. Instead of adding traceability to keys, white-box cryptography can be used to force a traitor to distribute the entire decryption implementation (with embedded decryption key), providing a larger data set to implement traceability. A specific white-box instance was introduced by Billet and Gilbert [20], who proposed a new family of block ciphers with traceability capabilities, based on the Isomorphisms of Polynomials (IP) problem [145, 147, 50]. Unfortunately, the underlying building block was cryptanalyzed by Faugère and Perret [64], but a repair presented by Bringer *et al.* [31], based on a perturbation idea that was introduced to re-enforce the IP-based cryptosystems [60].

5.6 Conclusions

In this Chapter, we have described applications and research domains that might benefit from the white-box cryptography, and vice versa. We have shown that white-box cryptography can be used as a tool to create new cryptographic schemes. In particular, asymmetric schemes are of interest, since the obfuscation of a symmetric encryption function naturally yields an asymmetric behavior. However, as discussed in Chapter 4, this is not staightforward to achieve, since it must be careful investigated if cryptographic properties (e.g., security notions) remain maintained.

A slightly weaker attack model, is the side-channel attack model (see also Sect. 2.3.2). Solutions for white-box cryptography naturally offer protection techniques for circuits, when they are suitable to be implemented in hardware (there might be size and performance restrictions). We have elaborated on the use of

white-box cryptography for hardware devices.

The original strategy to white-box cryptography algorithms by transforming them into a encoded network of lookup tables, originates from *data flow obfuscation* research, where values in a program are computed with in encoded form. Implementing the subsequent decoding, computing and re-encoding of the values into small lookup tables is the preceding idea of white-boxing as proposed by Chow *et al.*. We elaborate on the use of white-box cryptography for computing on encoded values. We also make a connection towards secure function evaluation, another technique to compute on encoded values.

A fourth application domain we elaborate upon, is that of software protection. This includes techniques to prevent tampering of software implementations, by means of self-cheking techniques, or software hardening techniques. Within the context of networked applications, trustworthy execution is becoming a basic requirement. We discuss various technologies that are being developed, in order to offer a certain degree of guarantee to an entity that software is executed *trustworthy* on untrusted machines. One specific architecture that describes this setting is the White-Box Remote Program Execution model.

To conclude, we introduce the concept of *Digital Rights Management*, and discuss traitor tracing.

Chapter 6

Conclusions and Further Research

6.1 Conclusions

Cryptographic algorithms are tools to protect the information infrastructure in our rapidly evolving information society. We become increasingly dependent on the exchange of data, and the deployment of complex software applications, supported by the growing availability of communication networks and open computing platforms. The trend towards increased access to software applications offers new opportunities on the one hand, but poses new threats on the other hand.

In many applications, an adversary can obtain access to the implementation of cryptosystems, and reveal certain aspects of an algorithm, that eventually can lead to the extraction of confidential information. We have described a novel attack model, coined the white-box attack model, in which an adversary has access to the software implementation of cryptographic primitives, and administrative privileges on its execution platform.

In this thesis, we studied the subject of white-box cryptography, which addresses implementation issues of cryptographic primitives that are subject to white-box attacks. We have investigated three major approaches to the topic of white-box cryptography.

White-box implementations. The central failing of black-box security models is that the specific implementation of the software algorithm is considered to be irrelevant for security. However, when considered within a white-box attack

context, the choice of implementation is the only line of defense.

In [43, 42], Chow *et al.* introduced white-box cryptography and proposed white-box implementations for the DES and the AES respectively. These constructions were hard to evaluate and only indicative metrics were presented. In subsequent work, improved implementations were presented, and their robustness were investigated by a cryptanalytic process. We have presented the main cryptanalytic results, that consist of truncated differential fault injection attacks and algebraic attacks. In conclusion to this work, it appears that widely used block ciphers are weak against attacks on their software implementation, and it seems difficult to improve their strength. In particular, the algebraic attacks are surprisingly effective. Up to some extent, there even seems to be an orthogonality between traditional (black-box) design criteria for block ciphers, and white-box design criteria.

Hence, we suggest that a new family of block ciphers needs to be designed, which take white-box design criteria into consideration. Suggestions towards new constructions were formulated, based on generalized cryptanalysis results and analysis of basic building blocks.

Theoretical foundations. A theoretical model for white-box cryptography is presented, inspired by theoretical models for code obfuscation. WBC requires that some given scheme remains secure even if the adversary is given white-box access to a functionality instead of just black-box access. This is closely related to research in code obfuscation.

We present different models of code obfuscation and their respective theoretical results. Unfortunately, different notions of code-obfuscation imply distinct (im)possibility results. Moreover, it is difficult to fix one single suitable notion of obfuscation, that is neither too strong (such that positive results can be achieved) and neither too weak (such that results are meaningful).

We conceive a slightly different approach, and capture the requirements of white-box cryptography using a *white-box property (WBP)*. The WBP is defined for some cryptographic scheme and an appropriate security notion, and implies that with respect to the security notion, an obfuscation does not leak any "useful" information, even though it may leak some "useless" non-black-box information. This way, we aim to achieve a meaningful notion of obfuscation, and obtain positive results for obfuscators of *specific* classes of interesting programs (such as cryptographic primitives). The main result is a negative one – we show that for every non approximately learnable family, there exist certain (contrived) security notions for which the WBP fails. On the positive side, we show that under reasonable computational assumptions, there exist obfuscators that are able to implement cryptographic primitives in a secure way. E.g., an obfuscation of a symmetric encryption is presented, that is IND-CPA secure under the bilinear

Diffie-Hellman assumption.

Applications. The white-box attack model reflects the attack model of many practical applications. As a result, research in white-box cryptography is beneficial for a vast range of applications. This includes digital rights management applications, cryptographic software deployed on several devices such as mobile phones and set-top boxes, gaming platforms, and mobile agents. Moreover, techniques developed for white-box cryptography carry over to other topics related to cryptology, and vice versa. This includes the deployment of white-box techniques as a countermeasure for side-channel cryptanalysis, e.g., to obtain secure implementations for mobile devices. Also, while white-box implementations mainly focus on the secure implementation of standalone applications (e.g., implementation of encryption functions), we can extent the results towards a networked scenario. This is of interest within a context of trustworthy execution of software, and has its applications in the gaming industry and in grid computing.

6.2 Future Work

White-box cryptography is a novel, challenging research topic that aims for solutions in the most powerful attack model. It also has connections to a substantial number of other topics (e.g., secure function evaluation). We have explored three major approaches in this thesis, and have identified a broad range of future research directions. We present an assessment of the most important directions for future research.

Design of a new block cipher. In Chapter 3, we discussed the state of the art white-box implementations, and have presented cryptanalysis results on each of them. Similar to the research in block ciphers in the past decades, a process of scrutinizing needs to be initialized. New block ciphers that satisfy both black-box and white-box design criteria need to be presented, and analyzed in a public context. This should lead to improved analysis techniques, metrics, and design criteria. In Sect. 3.9.1, suggestions for the construction of such a block cipher are presented. One idea would consist in implementing lookup tables that use several non-compatible operators to thwart algebraic analysis, and make sure the implementation consists of lookup tables with small input size, while their output size can be arbitrary.

In the theoretical approach, we have presented some positive results on the existence of secure white-box implementations. However, the constructions are subject to discussion. The symmetric primitive that is obfuscated (and hence results into an asymmetric primitive), is deduced from a primitive that was orig-

inally asymmetric. This observation indicates that a new block ciphers could involve building blocks that are originally designed for asymmetric schemes, from areas including *Multivariate Cryptography* (*Hidden Field Equations (HFE)*, *Isomorphic Polymorphisms*, and C*; see Wolf [188]) for an overview.

Design criteria. Closely related to the study for a new block cipher, is the study on design criteria. These should go beyond metrics, and include design criteria obtained from cryptanalysis results. Preferable, up to some extent they should also agree with design criteria for block ciphers. The algebraic cryptanalysis results naturally extend from equivalence solving, due to the nature of the obfuscation process (injection of random annihilating encodings). Hence the research on equivalence solvers is an interesting related study subject. The following equivalence solvers have been used in algebraic cryptanalysis of white-box implementations (see also Sect. 3.6.2):

- Billet *et al.* [21]: extraction of the non-linear part of the encodings that have some (known) *double bijective* behaviour,
- Biryukov *et al.* [22]: linear and affine equivalence solvers for S-boxes,
- Michiels *et al.* [135]: equivalence solvers for matrics,
- Faugère *et al.* [64]: equivalence solvers for isomorphic polymorfisms.

Theoretical results. We have shown a positive result on obfuscation by presenting an obfuscatable encryption scheme that satisfies IND-CPA security, based on the bilinear Diffie-Hellman hardness assumption. An interesting direction is to formulate other meaningful constructions that can be shown secure for some security notion, such as a family of obfuscatable signature schemes that satisfies an appropriate security notion. A similar result has been obtained by Hofheinz *et al.* [91] under their notion of obfuscation, where the obfuscation of a *message authentication code (MAC)* that satisfies MAC-CMA (chosen message attack), results into a signature scheme that satisfies SIG-CMA.

A second interesting theoretical research direction is the reduction of WBP to IND-soundness. Such a simplified definition might be more appealing, however it is possible that IND-soundness is too strong to be satisfied. Remark that this direction is only interesting for the obfuscation of probabilistic functions. It has been shown (in Wee [186], Hofheinz *et al.* [91], and by Proposition 2) that deterministic functions satisfy IND-soundness when they are (approximately) learnable, hence ruling out interesting families of functions (such as cryptographic primitives). We also wish to emphasize that probabilistic (symmetric) primitives are also a much more interesting subject of study from a practical point

of view (where the randomness is part of the input), because their obfuscation then results into probabilistic asymmetric primitives (deterministic asymmetric primitives have some drawbacks).

Applications. In Chapter 5, we have presented some interesting directions for further research. These include directions towards the practical deployment of white-box techniques within the context of software protection (e.g., for the gaming industry), and the deployment on hardware and constrained devices; the development of new cryptographic primitives such as asymmetric schemes, programmable random oracles, and homomorphic encryption schemes; augmenting white-box techniques to enable extra functionality such as traitor tracing, software tamper resistance, and watermarking; and the aid towards trustworthy execution, and virtualization.

Conversably, there are related application domains and research topics that might be beneficial for white-box techniques such as masking techniques against side-channel cryptanalysis, asymmetric cryptosystems (e.g., multivariate cryptography), and techniques for secure function evaluation.

Bibliography

[1] Carlisle M. Adams and Stafford E. Tavares. Designing S-boxes for ciphers resistant to differential cryptanalysis. In *Proceedings of the 3rd Symposium on State and Progress of Research in Cryptography*, pages 181–190, 1993. 18

[2] Ben Adida and Douglas Wikström. How to Shuffle in Public. In *Proceedings of 4th Theory of Cryptography Conference (TCC 2007)*, volume 4392 of *Lecture Notes in Computer Science*, pages 555–574. Springer-Verlag, 2007. 95

[3] Ibrahim A. Al-Kadi. *The origins of cryptology: The Arab contributions*, volume 16(2) of *Cryptologia*, pages 97–126. April 1992. 4

[4] ATMExpress. Understanding ATM Security: Tripe DES Technology, Remote Key Entry, and EPP's. https://www.atmexpress.com/downloads/tripledes.pdf. 21

[5] Boaz Barak. How to Go Beyond the Black-Box Simulation Barrier. In *Proceedings of the 42nd symposium on Foundations of Computer Science (FOCS 2001)*, IEEE Computer Society, pages 106–115, Washington, DC, USA, 2001. IEEE Computer Society. 86

[6] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (Im)possibility of Obfuscating Programs. In *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18. Springer-Verlag, 2001. 81, 84, 85, 86, 87, 89, 90, 91, 93, 95, 96, 100, 103, 104, 110, 124, 136

[7] P. Barreto and V. Rijmen. The Khazad legacy-level block cipher. In *First open NESSIE Workshop*, page 15. 13-14 November 2000. 62

[8] Mihir Bellare, Anand Desai, E. Jokipii, and Phillip Rogaway. A Concrete Security Treatment of Symmetric Encryption. In *Proceedings of the 38th Symposium on Foundations of Computer Science (FOCS 1997)*, IEEE Computer Society, pages 394–403, 1997. 97, 98, 99

[9] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. In *Advances in Cryptology - CRYPTO 1998*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer-Verlag, 1998. 96, 97

[10] Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security (CCS 1993)*, pages 62–73. ACM Press, 1993. 84, 93, 125

[11] Mihir Bellare and Phillip Rogaway. The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer-Verlag, 2006. 96, 97

[12] Ryad Benadjila, Olivier Billet, and Stanislas Francfort. Drm to counter side-channel attacks? In *Proceedings of 7th ACM Workshop on Digital Rights Management (DRM 2007)*, pages 23–32, New York, NY, USA, 2007. ACM Press. 128

[13] Eli Biham. New types of cryptanalytic attacks using related keys. In *Advances in Cryptology - EUROCRYPT 1993*, volume 765 of *Lecture Notes in Computer Science*, pages 398–409, Secaucus, NJ, USA, 1994. Springer-Verlag. 28

[14] Eli Biham. Cryptanalysis of Patarin's 2-Round Public Key System with S Boxes (2R). In *Advances in Cryptology - EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 408–416. Springer-Verlag, 2000. 41

[15] Eli Biham, Ross J. Anderson, and Lars R. Knudsen. Serpent: A New Block Cipher Proposal. In *Proceedings of the 5th International Workshop on Fast Software Encryption (FSE 1998)*, volume 1372 of *Lecture Notes in Computer Science*, pages 222–238, London, UK, 1998. Springer-Verlag. 62

[16] Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In *Advances in Cryptology - CRYPTO 1990*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer-Verlag, 1990. 21, 27

[17] Eli Biham and Adi Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. Springer-Verlag, London, UK, 1993. 27

[18] Eli Biham and Adi Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In *Advances in Cryptology - CRYPTO 1997*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer-Verlag, 1997. 55

[19] Eli Biham and Adi Shamir. Power Analysis of the Key Scheduling of the AES Candidates. Presented at the 2nd AES Candidate Conference, Rome, March 22–23, 1999. 36, 37

[20] Olivier Billet and Henri Gilbert. A Traceable Block Cipher. In *Advances in Cryptology - ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 331–346. Springer-Verlag, 2003. 41, 63, 81, 141

[21] Olivier Billet, Henri Gilbert, and Charaf Ech-Chatbi. Cryptanalysis of a White Box AES Implementation. In *Proceedings of the 11th International Workshop on Selected Areas in Cryptography (SAC 2004)*, volume 3357 of *Lecture Notes in Computer Science*, pages 227–240. Springer-Verlag, 2004. 44, 59, 60, 65, 82, 83, 146

[22] Alex Biryukov, Christophe De Cannière, An Braeken, and Bart Preneel. A Toolbox for Cryptanalysis: Linear and Affine Equivalence Algorithms. In *Advances in Cryptology - EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 33–50. Springer-Verlag, 2003. 61, 62, 65, 146

[23] Dan Boneh, Richard A. Demillo, and Richard J. Lipton. On the importance of eliminating errors in cryptographic computations. *Journal of Cryptology*, 14(2):101–119, 2001. 36, 37

[24] Dan Boneh and Matthew K. Franklin. Identity-Based Encryption from the Weil Pairing. In *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer-Verlag, 2001. 113, 114, 115

[25] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In *Advances in Cryptology - EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer-Verlag, 2003. 113

[26] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF Formulas on Ciphertexts. In *Proceedings of 2th Theory of Cryptography Conference (TCC 2005)*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer-Verlag, 2005. 40, 129

[27] Dan Boneh, Ben Lynn, and Hovav Shacham. Short Signatures from the Weil Pairing. In *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532, London, UK, 2001. Springer-Verlag. 113

[28] Dan Boneh and James Shaw. Collusion-Secure Fingerprinting for Digital Data (Extended Abstract). In *Advances in Cryptology - CRYPTO 1995*, volume 963 of *Lecture Notes in Computer Science*, pages 452–465, London, UK, 1995. Springer-Verlag. 141

[29] Johan Borst, Bart Preneel, and V Joos. On the time-memory tradeoff between exhaustive key search and table precomputation. In *Proceedings of the 19th Symposium in Information Theory in the Benelux, WIC*, pages 111–118, 1998. 26

[30] An Braeken. *Cryptographic Properties of Boolean Functions and S-Boxes*. PhD thesis, Katholieke Universiteit Leuven, 2006. 80

[31] Julien Bringer, Hervé Chabanne, and Emmanuelle Dottax. Perturbing and Protecting a Traceable Block Cipher. In *Proceedings of the 10th Communications and Multimedia Security (CMS 2006)*, volume 4237 of *Lecture Notes in Computer Science*, pages 109–119. Springer-Verlag, 2006. 41, 141

[32] Julien Bringer, Herve Chabanne, and Emmanuelle Dottax. White box cryptography: Another attempt. Cryptology ePrint Archive, Report 2006/468, 2006. http://eprint.iacr.org/. 41, 81

[33] Lawrence Brown and Jennifer Seberry. On the design of permutation P in DES type cryptosystems. In *Advances in Cryptology - EUROCRYPT 1989*, volume 434 of *Lecture Notes in Computer Science*, pages 696–705, New York, NY, USA, 1990. Springer-Verlag. 74

[34] Ran Canetti. Towards Realizing Random Oracles: Hash Functions That Hide All Partial Information. In *Advances in Cryptology - CRYPTO 1997*, volume 1294 of *Lecture Notes in Computer Science*, pages 455–469, London, UK, 1997. Springer-Verlag. 86, 93, 94, 96, 117

[35] Ran Canetti and Ronny Ramzi Dakdouk. Obfuscating Point Functions with Multibit Output. In *Advances in Cryptology - EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 489–508. Springer-Verlag, 2008. 84, 94, 112, 126

[36] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *Journal of the ACM*, 51(4):557–594, 2004. 126

[37] Ran Canetti, Daniele Micciancio, and Omer Reingold. Perfectly One-Way Probabilistic Hash Functions (Preliminary Version). In *Proceedings of the 30th ACM Symposium on Theory of Computing (STOC 1998)*, pages 131–140. ACM Press, 1998. 93, 96

[38] Jan Cappaert, Brecht Wyseur, and Bart Preneel. Software security techniques. COSIC internal report, Katholieke Universiteit Leuven, 2004. 86, 130

[39] Vincent Carlier, Herve Chabanne, and Emmanuelle Dottax. Grey-Box Implementation of Block Ciphers Preserving the Confidentiality of their Design. Cryptology ePrint Archive, Report 2004/188, 2004. http://eprint.iacr.org/. 41

[40] Suresh Chari, Charanjit Jutla, Josyula R. Rao, and Pankaj Rohatgi. A cautionary note regarding evaluation of AES candidates on smart-cards. In *2nd Advanced Encryption Standard Candidate Conference*, pages 133–147, 1999. 36, 37

[41] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template Attacks. In *Proceedings of 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2002)*, volume 2523, pages 13–28, London, UK, 2003. Springer-Verlag. 30

[42] Stanley Chow, Philip A. Eisen, Harold Johnson, and Paul C. van Oorschot. White-Box Cryptography and an AES Implementation. In *Proceedings of the 9th International Workshop on Selected Areas in Cryptography (SAC 2002)*, volume 2595 of *Lecture Notes in Computer Science*, pages 250–270. Springer, 2002. xxiv, 14, 34, 36, 40, 43, 44, 45, 50, 51, 54, 61, 83, 144

[43] Stanley Chow, Philip A. Eisen, Harold Johnson, and Paul C. van Oorschot. A white-box DES implementation for DRM applications. In *Proceedings of the ACM Workshop on Security and Privacy in Digital Rights Management (DRM 2002)*, volume 2696 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2002. xxiv, 14, 36, 38, 40, 43, 44, 45, 46, 50, 54, 61, 77, 83, 140, 144

[44] Cloakware. Cloakware Introduces High-Security Software Solutions for North American Cable Operators. http://security.cloakware.com/news/press-releases-details.php?id=116, December 3 2008. 140

[45] Christian Collberg, Clark Thomborson, and Douglas Low. A Taxonomy of Obfuscating Transformations. Technical Report 148, July 1997. 43, 85

[46] Christian Collberg, Clark Thomborson, and Douglas Low. Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs. In *Principles of Programming Languages (POPL 1998)*, San Diego, CA, January 1998. 43

[47] Christian S. Collberg, Clark D. Thomborson, and Douglas Low. Breaking Abstractions and Unstructuring Data Structures. In *Proceedings of the 1998 International Conference on Computer Languages (ICCL 1998)*, IEEE Computer Society, pages 28–38, 1998. 11

[48] D. Coppersmith. The Data Encryption Standard (DES) and its strength against attacks. *IBM J. Res. Dev.*, 38(3):243–250, 1994. 72

[49] IBM Corporation. IBM Systems Virtualization - description of basic concepts. available online at http://publib.boulder.ibm.com/infocenter/eserver/v1r2/topic/eicay/eicay.pdf, 2005. 133

[50] Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In *Advances in Cryptology - EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407. Springer-Verlag, 2000. 41, 141

[51] Nicolas Courtois and Josef Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In *Advances in Cryptology - ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287, London, UK, 2002. Springer-Verlag. 28

[52] Joan Daemen, Michael Peeters, and Gilles Van Assche. Bitslice Ciphers and Power Analysis Attacks. In *Proceedings of the 7th International Workshop on Fast Software Encryption (FSE 2000)*, volume 1978 of *Lecture Notes in Computer Science*, pages 134–149, London, UK, 2001. Springer-Verlag. 36, 37

[53] Joan Daemen and Vincent Rijmen. Resistance against implementation attacks: a comparative study of the AES proposals. In *Proceedings of the 2nd AES Candidate Conference*, pages 122–132, 1999. 36, 37

[54] Joan Daemen and Vincent Rijmen. *The Design of Rijndael*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002. 23, 25

[55] Alexander W. Dent. Fundamental problems in provable security and cryptography. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 364(1849):3215–3230, 2006. 96

[56] Yvo Desmedt and Jean-Jacques Quisquater. Public-Key Systems Based on the Difficulty of Tampering (Is There a Difference Between DES and RSA?). In *Advances in Cryptology - CRYPTO 1986*, volume 263 of *Lecture Notes in Computer Science*, pages 111–117. Springer-Verlag, 1987. 124

[57] Yvo Desmedt, Jean-Jacques Quisquater, and M. Davio. Dependence of output on input in DES. Small avalanche characteristics. In *Advances in Cryptology - CRYPTO 1984*, volume 196 of *Lecture Notes in Computer Science*, pages 359–376. Springer-Verlag, 1985. 57

[58] Leonard Eugene Dickson. *Linear Groups with an Exposition of the Galois Field Theory*. New York: Dover Publications, 1958. With an introduction by Wilhelm Magnus. 43

[59] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976. 16, 124

[60] Jintai Ding. A New Variant of the Matsumoto-Imai Cryptosystem through Perturbation. In *Proceedings of the 7th International Workshop on Theory and Practice in Public Key Cryptography (PKC 2004)*, volume 2947 of *Lecture Notes in Computer Science*, pages 305–318. Springer-Verlag, 2004. 41, 141

[61] Yevgeniy Dodis and Adam Smith. Correcting errors without leaking partial information. In *Proceedings of the 37th ACM Symposium on Theory of Computing (STOC 2005)*, pages 654–663. ACM Press, 2005. 94

[62] Electronic Frontier Foundation DES Cracker Press Release, July 17, 1998. http://w2.eff.org/Privacy/Crypto/Crypto_misc/DESCracker/HTML/19980716_eff_descracker_pressrel.html. 21

[63] Engadget. FairUse4WM strips Windows Media DRM! http://www.engadget.com/2006/08/25/fairuse4wm-strips-windows-media-drm, August 25 2006. 9

[64] Jean-Charles Faugère and Ludovic Perret. Polynomial Equivalence Problems: Algorithmic and Theoretical Aspects. In *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 30–47. Springer-Verlag, 2006. 41, 63, 141, 146

[65] Sebastian Faust, Brecht Wyseur, and Gregory Neven. Pin-Based Digital Lockers. COSIC internal report, 2007. 94

[66] Ed Felten. AACS: Extracting and Using Keys. http://www.freedom-to-tinker.com/?p=1106, January 10, 2007. 9, 140

[67] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic Analysis: Concrete Results. In *Proceedings of 3rd International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2001)*, volume 2162, pages 251–261, London, UK, 2001. Springer-Verlag. 30

[68] Juan A. Garay and Lorenz Huelsbergen. Software integrity protection using timed executable agents. In *Proceedings of the ACM Symposium on Information, computer and communications security (ASIACCS 2006)*, pages 189–200, New York, NY, USA, 2006. ACM. 135

[69] Craig Gentry. On Homomorphic Encryption over Circuits over Arbitrary Depth. In *Proceedings of the 41th ACM Symposium on Theory of Computing (STOC 2009)*, New York, NY, USA, 2009. ACM Press. 129

[70] Jonathon T. Giffin, Mihai Christodorescu, and Louis Kruger. Strengthening Software Self-Checksumming via Self-Modifying Code. In *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC 2005)*, pages 23–32, Washington, DC, USA, 2005. IEEE Computer Society. 131

[71] Oded Goldreich. A uniform complexity treatment of encryption and zero-knowledge. *Journal of Cryptology*, 6:21–53, 1993. 98

[72] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, New York, NY, USA, 2000. 23, 84

[73] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In *Proceedings of the 19th ACM Symposium on Theory of Computing (STOC 1987)*, pages 218–229. ACM Press, 1987. 40, 130

[74] S Goldwasser, S Micali, and C Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the 17th annual ACM Symposium on Theory of Computing (STOC 1985)*, pages 291–304, New York, NY, USA, 1985. ACM Press. 86

[75] Shafi Goldwasser and Mihir Bellare. Lecture Notes on Cryptography. `http://www-cse.ucsd.edu/~mihir/papers/gb.html`, August 2001. 283 pages. 84

[76] Shafi Goldwasser and Yael Tauman Kalai. On the Impossibility of Obfuscation with Auxiliary Input. In *Proceedings of the 46th Symposium on Foundations of Computer Science (FOCS 2005)*, IEEE Computer Society, pages 553–562, Washington, DC, USA, 2005. IEEE Computer Society. 84, 91, 93, 96, 100, 103, 140

[77] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. One-Time Programs. In *Advances in Cryptology - CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 39–56. Springer-Verlag, 2008. 127

[78] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information. In *Proceedings of the 14th ACM Symposium on Theory of Computing (STOC 1982)*, pages 365–377. ACM Press, 1982. 96, 97, 128

[79] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. 41, 96, 97

[80] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988. 97

[81] Shafi Goldwasser and Guy N. Rothblum. On Best-Possible Obfuscation. In *Proceedings of 4th Theory of Cryptography Conference (TCC 2007)*, volume 4392 of *Lecture Notes in Computer Science*, pages 194–213. Springer-Verlag, 2007. 84, 93

[82] Louis Goubin, Jean-Michel Masereel, and Michaël Quisquater. Cryptanalysis of White Box DES Implementations. In *Proceedings of the 14th International Workshop on Selected Areas in Cryptography (SAC 2007)*, volume 4876 of *Lecture Notes in Computer Science*, pages 278–295. Springer-Verlag, 2007. 44, 54, 57, 68, 83

[83] David Grawrock. *The Intel Safer Computing Initiative – Building Blocks for Trusted Computing*. Intel Press, 2006. 133

[84] Satoshi Hada. Zero-Knowledge and Code Obfuscation. In Tatsuaki Okamoto, editor, *Advances in Cryptology - ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 443–457, London, UK, 2000. Springer-Verlag. 84, 86, 89, 91, 92

[85] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. Lest we remember: cold boot attacks on encryption keys. In *Proceedings of the 17th USENIX Security Symposium (USENIX 2008)*, pages 45–60, 2008. 10

[86] Colin G. Harrison, David M. Chess, and Aaron Kershenbaum. Mobile Agents: Are they a good idea? Technical report, IBM T.J. Watson Research Center, P.O.Box 704, Yorktown Heights, NY 10598, 1985. 7

[87] M. Hellman. A cryptanalytic time-memory trade off. *IEEE Transactions on Information Theory*, IT–26(4):401–406, 1980. 26

[88] Amir Herzberg and Haya Shulman. Robust combiners for white-box security. Cryptology ePrint Archive, Report 2008/150, 2008. http://eprint.iacr.org/. 137

[89] Amir Herzberg, Haya Shulman, Amitabh Saxena, and Bruno Crispo. Towards a theory of white-box security. Cryptology ePrint Archive, Report 2008/087, 2008. http://eprint.iacr.org/. 84, 95, 136, 137

[90] Hex-Rays. The IDA Pro Dissasembler and Debugger. http://www.hex-rays.com/idapro/. 11, 34

[91] Dennis Hofheinz, John Malone-Lee, and Martijn Stam. Obfuscation for Cryptographic Purposes. In *Proceedings of 4th Theory of Cryptography Conference (TCC 2007)*, volume 4392 of *Lecture Notes in Computer Science*, pages 214–232. Springer-Verlag, 2007. 84, 88, 89, 91, 92, 94, 95, 96, 100, 103, 112, 113, 117, 119, 124, 146

[92] Greg Hoglund and Gary McGraw. *Exploiting online games: cheating massively distributed systems*. Addison-Wesley Professional, 2007. 135

[93] Susan Hohenberger, Guy Rothblum, Abhi Shelat, and Vinod Vaikuntanathan. Securely Obfuscating Re-Encryption. In *Proceedings of 4th Theory of Cryptography Conference (TCC 2007)*, volume 4392 of *Lecture Notes in Computer Science*, pages 233–252. Springer-Verlag, 2007. 37, 84, 87, 89, 92, 94, 95, 96, 100, 103, 104, 112, 115, 117, 119, 120, 124

[94] Fritz Hohl. Time Limited Blackbox Security: Protecting Mobile Agents From Malicious Hosts. In *Mobile Agents and Security*, volume 1419, pages 92–113, London, UK, 1998. Springer-Verlag. 7, 34, 44, 83

[95] Bill Horne, Lesley R. Matheson, Casey Sheehan, and Robert Endre Tarjan. Dynamic self-checking techniques for improved tamper resistance. In *DRM '01: Revised Papers from the ACM CCS-8 Workshop on Security and Privacy in Digital Rights Management*, pages 141–159, London, UK, 2002. Springer-Verlag. 131

[96] Russell Impagliazzo and Michael Luby. One-way Functions are Essential for Complexity Based Cryptography (Extended Abstract). In *Proceedings of the 30th Symposium on Foundations of Computer Science (FOCS 1989)*, IEEE Computer Society, pages 230–235, 1989. 91

[97] Apple Inc. iTunes home page. http://www.apple.com/itunes/. 8, 139

[98] Intel. Advanced Encryption Standard (AES) Instruction Set. http://software.intel.com/en-us/articles/advanced-encryption-standard-aes-instructions-set, August 25 2008. 23

[99] Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private Circuits II: Keeping Secrets in Tamperable Circuits. In *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 308–327. Springer-Verlag, 2006. 30

[100] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *In Proceedings of CRYPTO 2003*, pages 463–481. Springer-Verlag, 2003. 30

[101] Matthias Jacob, Dan Boneh, and Edward W. Felten. Attacking an Obfuscated Cipher by Injecting Faults. In *Proceedings of the ACM Workshop on Security and Privacy in Digital Rights Management (DRM 2002)*, volume 2696 of *Lecture Notes in Computer Science*, pages 16–31. Springer, 2002. 44, 50, 54, 55

[102] Marc Joye. On White-Box Cryptography. In Atilla Elci, S. Berna Ors, and Bart Preneel, editors, *First International Conference on Security of Information and Networks (SIN 2007)*, Security of Information and Networks, pages 7–12. Trafford Publishing, May 7-11 2007. 124

[103] David Kahn. *The Codebreakers: The Story of Secret Writing*. New York: Macmillan, 1967. 3, 4, 5

[104] Auguste Kerckhoffs. La cryptography militaire. *Journal des sciences militaires*, IX:5–38, Janvier 1883. 4, 16

[105] Tim Kerins and Klaus Kursawe. A cautionary note on weak implementations of block ciphers. In *1st Benelux Workshop on Information and System Security (WISSec 2006)*, page 12, Antwerp, BE, 2006. 11

[106] Joe Kilian and Phillip Rogaway. How to Protect DES Against Exhaustive Key Search. In *Advances in Cryptology - CRYPTO 1996*, volume 1109 of *Lecture Notes in Computer Science*, pages 252–267, London, UK, 1996. Springer-Verlag. 12

[107] Jongsung Kim, Seokhie Hong, and Bart Preneel. Related-Key Rectangle Attacks on Reduced AES-192 and AES-256. In *Proceedings of the 14th International Workshop on Fast Software Encryption (FSE 2007)*, volume 4593 of *Lecture Notes in Computer Science*. Springer-Verlag, 2007. 28

[108] Lars R. Knudsen. Truncated and Higher Order Differentials. In *Proceedings of the 2nd International Workshop on Fast Software Encryption (FSE 1994)*, volume 1008 of *Lecture Notes in Computer Science*, pages 196–211. Springer-Verlag, 1994. 27

[109] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Advances in Cryptology - CRYPTO 1996*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113, London, UK, 1996. Springer-Verlag. 28, 29

[110] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *Advances in Cryptology - CRYPTO 1999*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397, London, UK, 1999. Springer-Verlag. 29

[111] Vladimir Kolesnikov. Gate evaluation secret sharing and secure one-round twoparty computation. In *Advances in Cryptology - ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 136–155. Springer-Verlag, 2005. 40, 130

[112] Boris Köpf and David Basin. An information-theoretic model for adaptive side-channel attacks. In *Proceedings of the 14th ACM conference on Computer and communications security (CCS 2007)*, pages 286–296, New York, NY, USA, 2007. ACM Press. 30

[113] Advanced Access Content System Licensing Administrator (AACS LA). AACS – Advances Access Content System. http://www.aacsla.com/. 9

[114] RSA Laboratories. Pkcs #5 v2.1: Password-based cryptography standard. http://www.rsa.com/rsalabs/node.asp?id=2127, October 5, 2006. 94

[115] Xuejia Lai and James L. Massey. A Proposal for a New Block Encryption Standard. In *Advances in Cryptology - EUROCRYPT 1990*, volume 473 of *Lecture Notes in Computer Science*, pages 389–404. Springer-Verlag, 1990. 81

[116] Alan C. Lin. Software Obfuscation with Symmetric Cryptography. Master's thesis, Air Force Institute of Technology (AFIT), Wright-Patterson Air Force Base, Ohio, 2008. 44

[117] Yehuda Lindell and Benny Pinkas. An Efficient Protocol for Secure Two-Party Computation in the Presence of Malicious Adversaries. In *Advances in Cryptology - EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 52–78. Springer-Verlag, 2007. 40, 130

[118] Hamilton E. Link and William D. Neumann. Clarifying Obfuscation: Improving the Security of White-Box DES. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC 2005)*, volume 1, pages 679–684, Washington, DC, USA, 2005. IEEE Computer Society. 44, 45, 46, 48, 50, 54, 56

[119] Richard Lipton and Tomas Sander. An Additively Homomorphic Encryp- tion Scheme or How to Introduce a Partial Trapdoor in the Discrete Log, November 1997. 41

[120] Michael Luby and Charles Rackoff. How to Construct Pseudorandom Permutations from Pseudorandom Functions. *SIAM Journal on Computing*, 17(2):373–386, 1988. 23

[121] Dennis Luciano and Gordon Prichett. Cryptology: From caesar ciphers to public-key cryptosystems. *The College Mathematics Journal, Vol*, 18:2–17, 1987. 3

[122] B. Lynn, M. Prabhakaran, and A. Sahai. Positive Results and Techniques for Obfuscation. In *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 20–39. Springer-Verlag, 2004. 84, 93, 96, 103, 112, 117, 126, 139, 140

[123] A. Main and Paul C. van Oorschot. Software Protection and Application Security: Understanding the Battleground. In *International Course on State of the Art and Evolution of Computer Security and Industrial Cryptography*, volume LNCS, June 2003. 34

[124] Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology - EUROCRYPT 1993*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397, Secaucus, NJ, USA, 1994. Springer-Verlag. 21, 26, 27

[125] Mitsuru Matsui. The First Experimental Cryptanalysis of the Data Encryption Standard. In *Advances in Cryptology - CRYPTO 1994*, volume 839 of *Lecture Notes in Computer Science*, pages 1–11, London, UK, 1994. Springer-Verlag. 27

[126] Mitsuru Matsui and Atsuhiro Yamagishi. A New Method for Known Plaintext Attack of FEAL Cipher. In *Advances in Cryptology - EUROCRYPT 1992*, Lecture Notes in Computer Science. Springer-Verlag, 1992. 27

[127] Brian Matt, Andrew Reisse, Tom Van Vleck, Steve Schwab, and Patrick Leblanc. Self-protecting mobile agents obfuscation report - final report, network associates laboratories, report #03-015. Technical report, 2003. 7

[128] Gary McGraw and Greg Hoglund. Online games and security. *IEEE Security and Privacy*, 5(5):76–79, 2007. 135

[129] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1996. 5, 11, 17

[130] Ralph C. Merkle. Fast Software Encryption Functions. In *Advances in Cryptology - CRYPTO 1990*, volume 537 of *Lecture Notes in Computer Science*, pages 476–501, London, UK, 1991. Springer-Verlag. 12

[131] Thomas S. Messerges. Securing the AES Finalists Against Power Analysis Attacks. In *Proceedings of the 7th International Workshop on Fast Software Encryption (FSE 2000)*, volume 1978 of *Lecture Notes in Computer Science*, pages 150–164, London, UK, 2001. Springer-Verlag. 127

[132] Silvio Micali and Leonid Reyzin. Physically Observable Cryptography (Extended Abstract). In *Proceedings of the 1st Theory of Cryptography Conference (TCC 2004)*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer-Verlag, 2004. 30

[133] Sam Michiels, Kristof Verslype, Wouter Joosen, and Bart De Decker. Towards a software architecture for DRM. In *Proceedings of 5th ACM Workshop on Digital Rights Management (DRM 2005)*, pages 65–74. ACM Press, 2005. 140

[134] Wil Michiels and Paul Gorissen. Mechanism for software tamper resistance: an application of white-box cryptography. In *Proceedings of 7th ACM Workshop on Digital Rights Management (DRM 2007)*, pages 82–89. ACM Press, 2007. 132

[135] Wil Michiels, Paul Gorissen, and Henk D.L. Hollmann. Cryptanalysis of a Generic Class of White-Box Implementations. In *Proceedings of the 15th International Workshop on Selected Areas in Cryptography (SAC 2008)*, Lecture Notes in Computer Science. Springer-Verlag, 2008. 44, 61, 67, 79, 80, 146

[136] Microsoft. Microsoft Windows Media DRM home page. http://www.microsoft.com/windows/windowsmedia/forpros/drm/default.mspx. 8, 139

[137] Jasvir Nagra, Brecht Wyseur, and Thomas Herlea. Trust Model for Software And Hardware-based TR methods. RE-TRUST Deliverable D2.1/D3.1, 2007. 131

[138] National Institute of Standards and Technology: Advanced encryption standard. FIPS publication 197 (2001). http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf. 23, 62

[139] NIST. National Institute of Standards and Technology. http://www.nist.gov/. 21

[140] Roman Novak. SPA-Based Adaptive Chosen-Ciphertext Attack on RSA Implementation. In *Proceedings of the 5th International Workshop on Practice and Theory in Public Key Cryptosystems (PKC 2002)*, pages 252–262, London, UK, 2002. Springer-Verlag. 29

[141] Philippe Oechslin. Making a Faster Cryptanalytic Time-Memory Trade-Off. In *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 617–630, 2003. 26

[142] National Institute of Standards and Technology: Data Encryption Standard. FIPS publication 46-3 (1977). http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf. 21

[143] Rafail Ostrovsky and William E. Skeith III. Private Searching on Streaming Data. In *Advances in Cryptology - CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 223–240. Springer-Verlag, 2005. Full version appeared in Journal of Cryptology, 20(4):397-400, 2007. 95, 128

[144] Pascal Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in Cryptology - EUROCRYPT 1999*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer-Verlag, 1999. 40, 128

[145] Jacques Patarin. Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms. In *Advances in Cryptology - EUROCRYPT 1996*, volume 1070 of *Lecture Notes in Computer Science*, pages 33–48. Springer-Verlag, 1996. 41, 81, 141

[146] Jacques Patarin and Louis Goubin. Asymmetric cryptography with S-Boxes. In *Proceedings of the First International Conference on Information and Communication Security (ICICS 1997)*, pages 369–380, London, UK, 1997. Springer-Verlag. 40, 41

[147] Jacques Patarin, Louis Goubin, and Nicolas Courtois. Improved Algorithms for Isomorphisms of Polynomials. In *Advances in Cryptology - EUROCRYPT 1998*,

volume 1403 of *Lecture Notes in Computer Science*, pages 184–200. Springer-Verlag, 1998. 41, 141

[148] Charles P. Pfleeger. *Security in Computing*. Prentice-Hall, Englewood Cliffs, New-Jersey, 1989. 76

[149] Raphael C.-W. Phan. Related-Key Attacks on Triple-DES and DESX Variants. In *Topics in Cryptology - The Cryptographers' Track at RSA Conference (CT-RSA 2004)*, volume 2694 of *Lecture Notes in Computer Science*, pages 15–24. Springer-Verlag, 1994. 21

[150] ECRYPT Stream Cipher Project. AES-CTR benchmark performance. http://www.ecrypt.eu.org/stream/perf/pentium-m/benchmarks/aes-ctr/aes-128/. 53

[151] Jean-Jacques Quisquater and David Samyde. ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In *Proceedings of the International Conference on Research in Smart Cards (E-SMART 2001)*, pages 200–210, London, UK, 2001. Springer-Verlag. 30

[152] Michael O. Rabin. How to Exchange Secrets by Oblivious Transfer. Harvard Center for Research in Computer Technology, manuscript (1981). 40, 130

[153] Charles Rackoff and Daniel R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In *Advances in Cryptology - CRYPTO 1991*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer-Verlag, 1992. 97, 98

[154] RE-TRUST. Remote EnTrusting by RUn-time Software auThentication . http://www.re-trust.org. 134

[155] Oded Regev. Lattice-Based Cryptography. In *Advances in Cryptology - CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 131–141. Springer-Verlag, 2006. 129

[156] Vincent Rijmen and Bart Preneel. On weaknesses of non-surjective round functions. *Designs, Codes, and Cryptography*, 12:253–266, 1997. 20

[157] Ronald L. Rivest. Cryptography. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity (A)*, pages 717–755. 1990. 3

[158] Ahmad-Reza Sadeghi. private communication, March 15 2008. 134

[159] Ahmad-Reza Sadeghi and Christian Stüble. Property-based Attestation for Computing Platforms: Caring about properties, not mechanisms. In *New Security Paradigms Workshop 2004*, pages 67–77. ACM, 2004. 133

[160] Reiner Sailer, Xiaolan Zhang, Trent Jaeger, and Leendert van Doorn. Design and Implementation of a TCG-based Integrity Measurement Architecture. In *Proceedings of the 13th USENIX Security Symposium (USENIX 2004)*, pages 223–238, 2004. 133

[161] Tomas Sander and Christian F. Tschudin. On Software Protection via Function Hiding. In *Proceedings of the Second International Workshop on Information Hiding (IH 1998)*, volume 1525 of *Lecture Notes in Computer Science*, pages 111–123, London, UK, 1998. Springer-Verlag. 41

[162] Tomas Sander and Christian F. Tschudin. Protecting Mobile Agents Against Malicious Hosts. In *Mobile Agents and Security*, volume 1419 of *Lecture Notes in Computer Science*, pages 44–60, London, UK, 1998. Springer-Verlag. 7, 34, 44, 83

[163] Tomas Sander and Christian F. Tschudin. Towards Mobile Cryptography. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 215–224, 1998. 7, 34, 44, 83

[164] Tomas Sander, Adam Young, and Moti Yung. Non-Interactive CryptoComputing For NC^1. In *Proceedings of the 40th Symposium on Foundations of Computer Science (FOCS 1999)*, IEEE Computer Society, pages 554–567, 1999. 40, 130

[165] Amitabh Saxena and Brecht Wyseur. On White-Box Cryptography and Obfuscation. Cryptology ePrint Archive, Report 2008/273, 2008. http://eprint.iacr.org/. 84, 91, 124

[166] Dries Schellekens, Brecht Wyseur, and Bart Preneel. Remote attestation on legacy operating systems with trusted platform modules. In *1st International Workshop on Run Time Enforcement for Mobile and Distributed Systems (REM 2007)*, volume 197(1) of *Electronic Notes in Theoretical Computer Science*, pages 59–72, Dresden,Germany, 2008. Elsevier. 135, 136

[167] Werner Schindler, Kerstin Lemke, and Christof Paar. A Stochastic Model for Differential Side Channel Cryptanalysis. In *Proceedings of 7th International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2005)*, volume 3659 of *Lecture Notes in Computer Science*, pages 30–46. Springer-Verlag, 2005. 30

[168] Bruce Schneier. Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish). In *Proceedings of the International Workshop on Fast Software Encryption (FSE 1993)*, volume 809 of *Lecture Notes in Computer Science*, pages 191–204, London, UK, 1994. Springer-Verlag. 12

[169] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, and Chris Hall. On the Twofish Key Schedule. In *Proceedings of the 5th International Workshop on Selected Areas in Cryptography (SAC 1998)*, volume 1556 of *Lecture Notes in Computer Science*, pages 27–42. Springer-Verlag, 1998. 23

[170] Arvind Seshadri, Mark Luk, Elaine Shi, Adrian Perrig, Leendert van Doorn, and Pradeep K. Khosla. Pioneer: Verifying Code Integrity and Enforcing Untampered Code Execution on Legacy Systems. In *Proceedings of the 20th ACM Symposium on Operating Systems Principles 2005 (SOSP 2005)*, pages 1–16. ACM Press, 2005. 135

[171] Adi Shamir. On the Security of DES. In *Advances in Cryptology - CRYPTO 1985*, volume 218 of *Lecture Notes in Computer Science*, pages 280–281. Springer-Verlag, 1986. 27

[172] Adi Shamir and Nicko van Someren. Playing "Hide and Seek" with Stored Keys. In *Proceedings of the Third International Conference on Financial Cryptography (FC 1999)*, volume 1648 of *Lecture Notes in Computer Science*, pages 118–124. Springer-Verlag, 1999. xiii, 10

[173] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–, july, october 1948. 2, 4, 9, 15

[174] Claude E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4):656—-715, 1949. 18, 19

[175] Symantec. DRM and White-Box Cryptography. https://forums.symantec.com/syment/blog/article?message.uid=305948, June 26 2007. 140

[176] Sysersoft. Syser debugger. http://www.sysersoft.com/. 34

[177] Anne Tardy-Corfdir and Henri Gilbert. A Known Plaintext Attack of FEAL-4 and FEAL-6. In *Advances in Cryptology - CRYPTO 1991*, volume 576 of *Lecture Notes in Computer Science*. Springer-Verlag, 1992. 27

[178] The COPACOBANA project. http://www.copacobana.org/. 21

[179] W. Thompson. Cryptomorphic Programming: A Random Program Concept. Florida State University, C.S. Dept., Advanced Cryptography, 2005. 131

[180] Joachim Trescher and Paul Gorissen. Key distribution in unsafe environments. Philips research laboratories. 42

[181] A. M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proc. London Math. Soc.*, 2(42):230–265, 1936. 85

[182] Leslie G. Valiant. Universal Circuits (Preliminary Report). In *Proceedings of the 8th ACM Symposium on Theory of Computing (STOC 1976)*, pages 196–203. ACM Press, 1976. 130

[183] Paul C. van Oorschot, Anil Somayaji, and Glenn Wurster. Hardware-Assisted Circumvention of Self-Hashing Software Tamper Resistance. *IEEE Transactions on Dependable and Secure Computing*, 2(2):82–92, 2005. 13, 131, 132

[184] Paul C. van Oorschot and Michael J. Wiener. Improving Implementable Meet-in-the-Middle Attacks by Orders of Magnitude. In *Advances in Cryptology - CRYPTO 1996*, volume 1109 of *Lecture Notes in Computer Science*, pages 229–236, London, UK, 1996. Springer-Verlag. 28

[185] Kristof Verslype and Bart De Decker. A Flexible and Open DRM Framework. In *Proceedings of the 10th Communications and Multimedia Security (CMS 2006)*, volume 4237 of *Lecture Notes in Computer Science*, pages 173–184. Springer-Verlag, 2006. 140

[186] Hoeteck Wee. On Obfuscating Point Functions. In *Proceedings of the 37th ACM Symposium on Theory of Computing (STOC 2005)*, pages 523–532, New York, NY, USA, 2005. ACM Press. 84, 88, 89, 91, 92, 94, 95, 96, 100, 103, 104, 112, 117, 126, 146

[187] D. J. Wheeler and R. M. Needham. TEA, a tiny encryption algorithm. In *Proceedings of the 2nd International Workshop on Fast Software Encryption (FSE 1994)*, volume 1008 of *Lecture Notes in Computer Science*, pages 363–366. Springer-Verlag, 1995. 23

[188] Christopher Wolf. *Multivariate Quadratic Polynomials in Public Key Cryptography*. PhD thesis, Katholieke Universiteit Leuven, 2005. 81, 146

[189] Stefan Wolf. Unconditional Security in Cryptography. In *Lectures on Data Security, Modern Cryptology in Theory and Practice*, volume 1561 of *Lecture Notes in Computer Science*, pages 217–250. Springer-Verlag, July 1998. 5

[190] Brecht Wyseur, Mina Deng, and Thomas Herlea. A survey of homomorphic encryption schemes. COSIC internal report, Katholieke Universiteit Leuven, 2007. 40

[191] Brecht Wyseur, Wil Michiels, Paul Gorissen, and Bart Preneel. Cryptanalysis of White-Box DES Implementations with Arbitrary External Encodings. In *Proceedings of the 14th International Workshop on Selected Areas in Cryptography (SAC 2007)*, volume 4876 of *Lecture Notes in Computer Science*, pages 264–277. Springer-Verlag, 2007. 44, 54, 57, 58, 67, 79, 83

[192] Brecht Wyseur and Bart Preneel. Condensed White-Box Implementations. In *Proceedings of the 26th Symposium on Information Theory in the Benelux (BSIT 2005)*, pages 296–301, Brussels, Belgium, 2005. Werkgemeenschap voor Informatie- en Communicatietheorie. 46

[193] Brecht Wyseur, Karel Wouters, Mina Deng, Thomas Herlea, and Bart Preneel. On the design of a secure multimedia archive. In *1st Benelux Workshop on Information and System Security (WISSec 2006)*, page 14, Antwerp,BE, 2006. 140

[194] James Xiao and Yongxin Zhou. Generating large non-singular matrices over an arbitrary field with blocks of full rank. Cryptology ePrint Archive, Report 2002/096, 2002. http://eprint.iacr.org/. 51

[195] Hiroki Yamauchi, Yuichiro Kanzaki, Akito Monden, Masahide Nakamura, and Ken ichi Matsumoto. Software obfuscation from crackers' viewpoint. In *Proceedings of the 2nd IASTED international conference on Advances in computer science and technology (ACST 2006)*, pages 286–291, Anaheim, CA, USA, 2006. ACTA Press. 11

[196] Andrew Chi-Chih Yao. Protocols for Secure Computations (Extended Abstract). In *Proceedings of the 23rd Symposium on Foundations of Computer Science (FOCS 1982)*, IEEE Computer Society, pages 160–164, 1982. 40

[197] Frances F. Yao and Yiqun Lisa Yin. Design and Analysis of Password-Based Key Derivation Functions. In *Topics in Cryptology - The Cryptographers' Track at RSA Conference (CT-RSA 2005)*, volume 3376 of *Lecture Notes in Computer Science*, pages 245–261. Springer-Verlag, 2005. 94

[198] Dingfeng Ye, Kwok-Yan Lam, and Zong-Duo Dai. Cryptanalysis of "2R" Schemes. In *Advances in Cryptology - CRYPTO 1999*, volume 1666 of *Lecture Notes in Computer Science*, pages 315–325. Springer-Verlag, 1999. 41

Index

List of Publications

International articles

1. Amitabh Saxena, and Brecht Wyseur. On White-Box Cryptography and Obfuscation. Sumbitted to *IEEE Computer Security Foundations Symposium*, 2009.
2. Dries Schellekens, Brecht Wyseur, and Bart Preneel. Remote attestation on legacy operating systems with trusted platform modules. In *Science of Computer Programming*, volume 74(1-2), pages 13-22, 2008.
3. Dries Schellekens, Brecht Wyseur, and Bart Preneel. Remote attestation on legacy operating systems with trusted platform modules. In *Run Time Enforcement for Mobile and Distributed Systems*, volume 197(1) of *Electronic Notes in Theoretical Computer Science*, pages 59–72, Dresden,Germany, 2008. Elsevier.
4. Brecht Wyseur, Wil Michiels, Paul Gorissen, and Bart Preneel. Cryptanalysis of White-Box DES Implementations with Arbitrary External Encodings. In *Selected Areas in Cryptography*, volume 4876 of *Lecture Notes in Computer Science*, pages 264–277. Springer-Verlag, 2007.
5. Karel Wouters, Brecht Wyseur, and Bart Preneel. Security Model for a Shared Multimedia Archive. In *Automated Production of Cross Media Content for Multi-Channel Distribution*, IEEE Computer Society, pages 249–255, 2007.
6. Karel Wouters, Brecht Wyseur, and Bart Preneel. Lexical Natural Language Steganography Systems with Human Interaction. In *Information Warfare and Security*, pages 303–312, 2007.

National articles

7. Brecht Wyseur, Karel Wouters, Mina Deng, Thomas Herlea, and Bart Preneel. On the design of a secure multimedia archive. In *1st Benelux Workshop on Information and System Security (WISSec 2006)*, 14 pages, Antwerp, 2006.
8. Brecht Wyseur and Bart Preneel. Condensed White-Box Implementations. In *Proceedings of the 26th Symposium on Information Theory in the Benelux*, Werkgemeenschap voor Informatie- en Communicatietheorie, pages 296–301, 2005.

Invited presentations

9. Brecht Wyseur. Introduction to White-Box Cryptography and White-Box DES Implementations. ECRYPT Summer Course on Advanced Topics in Cryptography, Fodele, Crete, Greece, 2008.
10. Brecht Wyseur. White-Box Cryptography. ASCURE Academy Event, Brussels, Belgium, 2008.

Internal reports

11. Sebastian Faust, Brecht Wyseur, and Gregory Neven. PIN-based Digital Lockers. COSIC Internal report, 14 pages, Katholieke Universiteit Leuven, 2008.
12. Brecht Wyseur, Mina Deng, and Thomas Herlea. A Survey on Homomorphic Encryption Schemes. COSIC Internal report, 15 pages, Katholieke Universiteit Leuven, 2007.
13. Jan Cappaert, Brecht Wyseur, and Bart Preneel. Software Security Techniques. COSIC internal report, 42 pages, Katholieke Universiteit Leuven, 2004.

Project reports

14. Dries Schellekens and Brecht Wyseur. Comparative Analysis of RE-TRUST with Trusted Computing. RE-TRUST Deliverable D4.5, 2009.
15. Aldo Basile, Yoram Ofek, Alessandro Zorat, Brecht Wyseur, Jerome DAnoville; Igor Kotenko, and Paolo Falcarin. Trust and Security Analysis. RE-TRUST Deliverable D4.x, 2009.
16. Brecht Wyseur. Encrypted Code Report. RE-TRUST Deliverable D3.3, 2009.
17. Jerome D'Annoville, Brecht Wyseur, Dries Schellekens, Mariano Ceccato, and Stefano Di Carlo. First Analysis Encrypted Code and HW Assisted SW Protection. RE-TRUST Deliverable D3.2, 27 pages, 2008.
18. Brecht Wyseur, Jan Cappaert, Mariano Ceccato, and Stefano Di Carlo. Protection mechanisms for hardening the software application against analysis and tampering. RE-TRUST Deliverable D2.4, 16 pages, 2008
19. Jasvir Nagra, Brecht Wyseur, and Thomas Herlea. Trust Model for Software And Hardware-based TR methods. RE-TRUST Deliverable D2.1/D3.1, 12 pages, 2007.
20. Brecht Wyseur, Thomas Herlea, Dries Schellekens, and Jasvir Nagra. Hardware/ software-based method - initial architecture. RE-TRUST Deliverable D1.3, 9 pages, 2007.
21. Karel Wouters, and Brecht Wyseur. Security Model for a Multimedia Archive. IBBT/IPEA Deliverable D.4, 27 pages, 2007

Brecht Wyseur was born on September 26, 1981 in Ypres, Belgium. He received his degree of Master in Mathematics from K.U.Leuven, Belgium, in July 2003. His Masters' thesis dealt with the polynomial choice in the index calculus for the discrete logarithm problem. In August 2003, Brecht started working in the research group COSIC (Computer Security and Industrial Cryptography) at the Department of Electrical Engineering (ESAT) of the K.U.Leuven. The first four years of his research were sponsored by a grant from the IWT (Institute for the Promotion of Innovation by Science and Technology in Flanders).

Updates, resources, and further progress in my research on white-box cryptography can be followed on http://www.whiteboxcrypto.com/.