Doubly half-injective PRGs for incompressible white-box cryptography

Estuardo Alpirez Bock[1], Alessandro Amadori[2], Joppe W. Bos[3], Chris Brzuska[1], and Wil Michiels[2,3]

[1] Aalto University
{estuardo.alpirezbock,brzuska}@aalto.fi
[2] Technische Universiteit Eindhoven
A.Amadori@tue.nl
[3] NXP Semiconductors
{joppe.bos@nxp.com,wil.michiels}@nxp.com

Abstract. White-box cryptography was originally introduced in the setting of digital rights management with the goal of preventing a user from illegally re-distributing their software decryption program. In recent years, mobile payment has become a popular new application for white-box cryptography. Here, white-box cryptography is used to increase the robustness against external adversaries (i.e., not the user) who aim to misuse/attack the cryptographic functionalities of the payment application. A necessary requirement for secure white-box cryptography is that an adversary cannot extract the embedded secret key from the implementation. However, a white-box implementation needs to fulfill further security properties in order to provide useful protection of an application. In this paper we focus on the popular property *incompressibility* that is a mitigation technique against code-lifting attacks. We provide an incompressible white-box encryption scheme based on the standard-assumption of one-way permutations whereas previous works used either public-key type assumptions or non-standard symmetric-type assumptions.

Keywords: White-box cryptography, Incompressibility, One-way permutations

1 Introduction

White-box cryptography was introduced by Chow, Eisen, Johnson and van Oorschot in 2002 in order to protect keys in symmetric ciphers when implemented in insecure or adversarially controlled environments [10,9]. The original proposal was motivated by Digital Rights Management (DRM), and white-box cryptography has been used in this context for many years. In recent years, mobile payment applications became popular and, originally, relied on secure hardware that communicated via Near-Field Communication (NFC) (cf. NFC-based payment products by Mastercard, Visa and Google Wallet [38]). Android

This paper has been published by Springer in the proceedings of the CT-RSA Conference 2019 https://doi.org/10.1007/978-3-030-12612-4_10.

4.4 added host-card emulation (HCE) which allows to implement the NFC protocols in software-only. Hereby, white-box cryptography has become an integral building block of mobile payment applications. Mastercard promotes the use of white-box cryptography in the payment applications that Mastercard certifies. I.e., the Mastercard security guidelines for payment applications make the use of white-box cryptography *mandatory* [30].

The wide-spread deployment of white-box cryptography stands in contrast with the state-of-the-art in white-box research. Currently, there are no long-term secure white-box implementations of standard ciphers in the academic literature. Proposed white-box constructions for both DES [10,29] and AES [9,8,40,26] have been subsequently broken by [25,21,39] and [3,33,32,28], respectively. Moreover, Bos, Hubain, Michiels, and Teuwen [7] and Sanfelix, de Haas and Mune [36] introduced Differential Computational Analysis (DCA) which is a generic approach to extract emebedded keys from a large class of white-box implementations *automatically*, i.e., without human reverse-engineering effort. As explained in [31], popular frameworks for implementing white-box cryptography are particularly vulnerable to such automated attacks.

In order to promote research on good candidates for white-box cryptography, CHES 2017 organized the white-box competition CHES 2017 Capture the Flag Challenge [13] to white-box AES-128. Unfortunately, all candidates were broken eventually. Most candidates lasted only 2 days, whereas some candidates resisted attacks for several weeks. Such a level of short-term security might already be useful, as long as the secret key and the white-box design can be updated on a regular basis. In light of these results, one might wonder whether there exists a long-term secure white-box implementation of AES. Short of being able to provide a practically secure white-box implementation of AES itself, we approach feasibility from the reduction-based approach in cryptography and aim to base secure white-box implementations on well-studied, symmetric assumptions. Whereas attacks usually focus on key extraction, positive feasibility results should aim for stronger, more useful security notions.

Definitions. Systematic definitional studies of security properties for white-box cryptography have been undertaken by Delerablée, Lepoint, Paillier, and Rivain (DLPR [11]) and Saxena, Wyseur, Preneel (SWP [37]). Some of the early definitions have been revisited and refined subsequently [5,6,4]. Beyond the modest goal of security against key extraction, those works cover desirable asymmetry properties: A white-box *en*cryption program should not allow to decrypt (confidentiality), and a white-box *de*cryption program should not allow to encrypt (integrity).

While asymmetry is a desirable property (and, in particular, implies security against key extraction), in practice, code-lifting attacks are more prevalent: Given a software cryptographic implementation with an embedded secret key, the adversary might simply copy the complete implementation and run it on its own device without the need to recover the embedded secret key. As a means to mitigate code-lifting attacks (and subsequently re-distribution attacks) most

works discuss the notion of incompressibility. Additionally, DLPR also suggest traceability.

Incompressibility. Incompressibility aims to mitigate re-distribution attacks by building large-size white-box programs, which remain functional only in their complete form. As soon as the white-box program is compressed or fragments of the program are removed, the program loses its functionality. The intuitive justification of the usefulness of incompressibility is that if a decryption algorithm is several gigabytes large, then online re-distribution of that algorithm might not be feasible, reducing thus the chances of an adversary sharing the cryptographic code for unintended purposes. This approach is particularly useful for the case where one distributes a combination of software and hardware with large memory.

Constructions. DLPR and SWP show that public-key encryption schemes, considered as white-boxed symmetric encryption schemes, satisfy confidentiality. Interestingly, DLPR also show that the RSA function is incompressible when interpreted as a white-boxed cipher. Feasibility results are important, because they illustrate that the hardness of building a white-box version of AES does not hinge on a general impossibility of white-box encryption. In particular, the hardness of building a white-box version of AES is not subject to the general impossibility result for virtual black-box obfuscation shown in the seminal paper by Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan and Yang [1].

In a systematic analysis of the obstacles that white-box constructions for AES face, one might investigate the cryptographic tools and assumptions that are needed. At first sight, one might expect that white-boxing AES requires public-key type assumptions from Cryptomania (See Impagliazzo's survey on average-case complexity [22]) such as trapdoor functions. Indeed, if the white-boxed version of AES shall satisfy the same confidentiality guarantees as public-key encryption, then the oracle separation by Impagliazzo and Rudich [24] applies.[4]

In turn, for less demanding notions such as incompressibility, it is conceivable that white-boxing can be based on symmetric-key type MiniCrypt assumptions alone. Indeed, an important step in that direction was made in a recent work by Fouque, Karpman, Kirchner and Minaud (FKKM [14]). FKKM present a symmetric-style cipher (i.e., a cipher that looks like a genuine cipher-design and is not built on public-key type assumptions) and show that the cipher admits an incompressible implementation, based on a novel symmetric-style assumption.

In this work, we place feasibility of incompressible white-box cryptography fully in MiniCrypt. We provide a white-box encryption scheme and a white-box

[4] It applies conceptually in the sense that AES is a pseudorandom permutation which is a MiniCrypt primitive that is equivalent to the existence of one-way functions. Strictly speaking, the security of AES is a much stronger assumption than merely the assumption of a one-way function, but it is fair to conjecture that one cannot turn AES into a secure public-key encryption scheme without gaining insights into the question for how to build public-key encryption from one-way functions generally.

decryption scheme, whose incompressibility is based on the assumption of a one-way permutation (See Section 4 for a more detailed comparison between our construction and the construction by FKKM).

Summary of contribution. We contribute to the foundations of white-box cryptography by showing that incompressible white-box encryption and decryption schemes can be built based on the assumption of one-way permutations only thereby placing incompressible white-box cryptography fully in MiniCrypt.

Taking a step back, solid definitions as well as feasibility results and impossibility for white-box cryptography are needed to clarify whether it is realistic to pursue the goal of building white-box cryptography with useful long-term security properties, with reasonable efficiency, based on standard assumptions. As the CHES Capture the Flag Challenge 2017 demonstrates, providing a secure white-box implementation of AES is tremendously difficult, and thus obtaining a solid understanding of the feasibility and limits of white-box cryptography is needed rather urgently. Our results take a step towards such an understanding and we encourage further studies on the foundations of white-box cryptography.

2 Preliminaries and Notation

1^n denotes the security parameter in unary notation. Given a bit string x, we denote by $x[j:i]$ the bits j to i of the bit string x. end denotes the index of the last bit. By $a||b$ we denote the concatenation of two bit strings a and b. For a program P, we denote by $|P|$ its bit-size. We leave the choice of encoding of the program implicit in this work. We write oracles as superscript to the adversary $\mathcal{A}^{\mathsf{O}}$. All algorithms receive the security parameter 1^n as input. For ease of notation, we omit the security parameter for most of the article.

U_n denotes the uniform distribution over strings of length n. By $\leftarrow$, we denote the execution of a deterministic algorithm while $\leftarrow\!\!\$$ denotes the execution of a randomized algorithm. We denote by $:=$ the process of initializing a set, e.g. $S := \emptyset$, while $\leftarrow\!\!\$$ denotes the process of randomly sampling an element from a given set, e.g. $x \leftarrow\!\!\$ \{0,1\}^n$. When sampling x according to the probability distribution X, we denote the probability that the event $F(x) = 1$ happens by $\Pr_{x \leftarrow\!\!\$ X}[F(x)]$.

We sometimes use $\circ$ for function composition, i.e. $g \circ f(x)$ is the same as $g(f(x))$. For a natural number ℓ, we write $f^{\ell}(x)$ for $f \circ ... \circ f(x)$, where we apply f to x sequentially ℓ times. The latter notations are helpful to make terms easier to parse when a function is composed many times, as in the standard notation, each function application introduces a layer of brackets.

Definition 1. *A symmetric encryption scheme ξ consists of three polynomial-time algorithms* $(\texttt{Kgen}, \texttt{Enc}, \texttt{Dec})$ *such that* `Kgen` *and* `Enc` *are probabilistic polynomial-time algorithms (PPT), and* `Dec` *is deterministic. The algorithms have the syntax $k \leftarrow\!\!\$ \texttt{Kgen}(1^n)$, $c \leftarrow\!\!\$ \texttt{Enc}(1^n, k, m)$ and $m \leftarrow \texttt{Dec}(1^n, k, c)$. Moreover, the encryption scheme ξ satisfies* correctness*, i.e., for all messages $m \in$*

$\{0,1\}^*$,

$$\Pr[\mathtt{Dec}(k, \mathtt{Enc}(k, m)) = m] = 1 \tag{1}$$

where the probability is over the randomness of $\mathtt{Enc}$ *and* $k \leftarrow\!\!\$\ \mathtt{Kgen}(1^n)$.

Remark. To clarify wording (as scientific communities vary in their terminology), we consider a *cipher* a deterministic algorithm that is a *building block* for an encryption scheme, but is not an encryption scheme itself. That is, AES is a cipher, not an encryption scheme, while, e.g., AES-CBC or AES-GCM are symmetric encryption schemes.

We now include the definition of authenticated encryption. We use an indistinguishability definition of authenticated encryption that encodes both, the ciphertext integrity and the indistinguishability under chosen plaintext attacks (IND-CPA). Bellare and Namprempre [2] show that if a symmetric encryption scheme provides ciphertext integrity and IND-CPA security, then it is also indistinguishable under chosen ciphertext attacks (IND-CCA). We refer to their article as well as to Krawczyk [27] for more background on authenticated encryption.

Definition 2 (Authenticated encryption (AE)). *A symmetric encryption scheme* $\mathtt{se} = (\mathtt{AKgen}, \mathtt{AEnc}, \mathtt{ADec})$ *is an authenticated encryption scheme (AE-secure) if for all adversaries* $\mathcal{A}$*, the advantage*

$$\left|\Pr\left[\mathsf{EXP}_{AE}^{\mathcal{A},\mathtt{se}}(1^n) = 1\right] - \tfrac{1}{2}\right|$$

is negligible.

$\underline{\mathsf{EXP}_{AE}^{\mathcal{A},\mathtt{se}}(1^n)}$
$k \leftarrow\!\!\$\ \mathtt{AKgen}(1^n)$
$b \leftarrow\!\!\$\ \{0,1\}$
$b^* \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{ENC},\mathsf{DEC}}(1^n)$
return $(b = b^*)$

$\underline{\mathsf{ENC}(m)}$
if $b = 0$
  $c \leftarrow\!\!\$\ \mathtt{AEnc}(k, m)$
if $b = 1$
  $c \leftarrow\!\!\$\ \mathtt{AEnc}(k, 0^{|m|})$
$C \leftarrow C \cup \{c\}$
return c

$\underline{\mathsf{DEC}(c)}$
if $b = 0$
  if $c \notin C$
    $m \leftarrow \mathtt{ADec}(k, c)$
    return m
return $\perp$

2.1 Syntax of White-Box Cryptography

Definition 3 (White-Box Encryption Scheme). *A white-box encryption scheme* WBEnc *consists of four probabilistic polynomial-time algorithms* $(\mathtt{Kgen}, \mathtt{Enc}, \mathtt{Dec}, \mathtt{Comp})$*, where* $(\mathtt{Kgen}, \mathtt{Enc}, \mathtt{Dec})$ *is a symmetric encryption scheme and* $\mathtt{Comp}$ *is a publicly known (possibly) randomized compiling algorithm that takes as input the symmetric key* k *and generates a (probabilistic) white-box encryption algorithm* $\mathtt{Enc}_{\mathtt{WB}}$.

$$\mathtt{Enc}_{\mathtt{WB}} \leftarrow\!\!\$\ \mathtt{Comp}(k) \tag{2}$$

For all messages $m \in \{0,1\}^*$*, the randomized program* $\mathtt{Enc}_{\mathtt{WB}}(m)$ *produces a distribution that is statistically close to the distribution of the randomized program*

$\mathtt{Enc}(k, m)$. *Moreover, the following* correctness *property holds. For all messages* $m \in \{0,1\}^*$,

$$\Pr[\mathtt{Dec}(k, \mathtt{Enc}_{\mathtt{WB}}(m)) = m] = 1, \tag{3}$$

where the probability is over the randomness of $\mathtt{Enc}_{\mathtt{WB}}$ *and* $k \leftarrow\!\!\$\ \mathtt{Kgen}(1^n)$.

Remark. One can use $\mathtt{Enc}(k, \cdot)$ as well as $\mathtt{Enc}_{\mathtt{WB}}(\cdot)$ to encrypt a message under key k. Both programs require randomness, and an honest user can provide the program $\mathtt{Enc}_{\mathtt{WB}}(\cdot)$ with uniform randomness to generate a secure distribution of ciphertexts. We will not mention this feature again, as the security properties covered in this paper are concerned with the case that the owner of $\mathtt{Enc}_{\mathtt{WB}}(\cdot)$ misbehaves. Note that we only demand statistical closeness between $\mathtt{Enc}(k, \cdot)$ and $\mathtt{Enc}_{\mathtt{WB}}(\cdot)$ and not full functional equivalence, as notions such as traceability benefit from flexibility on the functionality requirement.

We now define a white-box decryption scheme analogously that produces a white-box of the decryption algorithm rather than the encryption algorithm. Note that in the case of white-box *encryption*, there is a ciphertext *distribution* for each message m. In turn, in the case of white-box *decryption*, for each ciphertext c, there is merely a single plaintext. Therefore, for white-box *decryption*, no requirement on statistical closeness is needed beyond correctness.

Definition 4 (White-Box Decryption Scheme). *A white-box decryption scheme WBDec consists of four probabilistic polynomial-time algorithms* ($\mathtt{Kgen}$, $\mathtt{Enc}, \mathtt{Dec}, \mathtt{Comp}$)*, where* ($\mathtt{Kgen}, \mathtt{Enc}, \mathtt{Dec}$) *is a symmetric encryption scheme and* $\mathtt{Comp}$ *is a publicly known (possibly) randomized compiling algorithm that takes as input the symmetric key* k *and generates a white-box* decryption *program* $\mathtt{Dec}_{\mathtt{WB}}$*, such that for all messages* $m \in \{0,1\}^*$,

$$\Pr[\mathtt{Dec}_{\mathtt{WB}}(\mathtt{Enc}(k, m)) = m] = 1, \tag{4}$$

where the probability is over the randomness of $k \leftarrow\!\!\$\ \mathtt{Kgen}(1^n)$, $\mathtt{Dec}_{\mathtt{WB}} \leftarrow\!\!\$\ \mathtt{Comp}(k)$ *and* $\mathtt{Enc}(k, \cdot)$.

3 Definitions

Incompressibility aims to make redistribution attacks harder by making the white-box program too large to distribute. The first formalization of incompressibility was given by DLPR, and the notion has been adopted and studied in several subsequent works [14,4,5]. We adopt the incompressibility notion by DLPR with minor modifications: DLPR consider deterministic ciphers, while we consider randomized encryption schemes. Therefore, our correctness requirement will ask to produce decryptable ciphertexts rather than ciphertexts that are equal to a target value, as can be defined for deterministic ciphers. Moreover, we will add an encryption oracle for sake of completeness. As the adversary has a white-box encryption algorithm, the adversary can emulate the encryption oracle up to statistical distance and thus, our modification is merely esthetic.

In the (δ, λ)-incompressibility game, conceptually, there are two collaborating adversaries. One is the adversary $\mathcal{A}$ that is given a white-box encryption program $\mathtt{Enc}_{\mathtt{WB}}$ and outputs some smaller value $\mathtt{Com}$. The second collaborating adversary is the decompression algorithm $\mathtt{Decomp}$ that will try to decompress $\mathtt{Com}$. The winning condition says that the pair of adversaries is successful if

(i) $\mathtt{Com}$ is shorter than $\mathtt{Enc}_{\mathtt{WB}}$ by λ bits *and*

(ii) the probability that the decompressed program $\mathtt{Decomp}(\mathtt{Com})$ produces a valid ciphertext (i.e., a ciphertext that decrypts correctly) for a random message $m \in \{0,1\}^n$ is greater than δ.

Definition 5 (Incompressibility). *A white-box encryption scheme WBEnc is INC-(δ, λ)-secure if for all PPT adversaries $\mathcal{A}$, the success probability*

$$\left|\Pr\left[\mathsf{EXP}^{\mathcal{A},\mathit{WBEnc}}_{\mathit{INC\text{-}(\delta,\,\lambda)}} = 1\right]\right|$$

is negligible, where the experiment $\mathsf{EXP}^{\mathcal{A},\mathit{WBEnc}}_{\mathit{INC\text{-}(\delta,\,\lambda)}}$ is defined as follows:

$\mathit{EXP}^{\mathcal{A},\mathit{WBEnc}}_{\mathit{INC\text{-}(\delta,\,\lambda)}}$

$k \leftarrow\!\!\$\ \mathtt{Kgen}(1^n)$
$\mathtt{Enc}_{\mathtt{WB}} \leftarrow\!\!\$\ \mathtt{Comp}(k)$
$\mathtt{Com} \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{RCA},\mathsf{ENC},\mathsf{DEC}}(\mathtt{Enc}_{\mathtt{WB}})$
if $\Pr_{m \leftarrow\!\!\$\ \{0,1\}^*}[\mathtt{Dec}(k, \mathtt{Decomp}(\mathtt{Com})(m)) = m] \geq \delta$
  and if $|\mathtt{Com}| \leq |\mathtt{Enc}_{\mathtt{WB}}| - \lambda$
  return 1
else return 0

$\mathsf{RCA}()$

$\mathtt{Enc}'_{\mathtt{WB}} \leftarrow\!\!\$\ \mathtt{Comp}(k)$
return $\mathtt{Enc}'_{\mathtt{WB}}$

$\mathsf{ENC}(m)$

$c \leftarrow\!\!\$\ \mathtt{Enc}(k, m)$
return c

$\mathsf{DEC}(c)$

$m \leftarrow \mathtt{Dec}(k, c)$
return m

Incompressibility for white-box decryption. The definition of incompressibility for white-box decryption is analogous to Definition 5, except that in the former, the compression attack targets a white-box decryption algorithm $\mathsf{WBDec}_{\mathtt{WB}}$ and thus, the winning condition is $\Pr_{m \leftarrow\!\!\$\ \{0,1\}^*}[\mathtt{Decomp}(\mathtt{Com})(\mathtt{Enc}(k, (m)) = m] \geq \delta$, where the randomness is over m and $\mathtt{Enc}$.

Definition 6. *A white-box decryption scheme WBDec is INC-(δ, λ)-secure if for all PPT adversaries $\mathcal{A}$, the advantage*

$$\left|\Pr\left[\mathsf{EXP}^{\mathcal{A},\mathit{WBDec}}_{\mathit{INC\text{-}(\delta,\,\lambda)}} = 1\right]\right|$$

is negligible, where the experiment $\mathsf{EXP}^{\mathcal{A},\mathit{WBDec}}_{\mathit{INC}\text{-}(\delta,\lambda)}$ *is defined as follows:*

$\mathsf{EXP}^{\mathcal{A},\mathit{WBDec}}_{\mathit{INC}\text{-}(\delta,\lambda)}$

$k \leftarrow\!\!\$\ \mathtt{Kgen}(1^n)$
$\mathtt{Dec}_{\mathtt{WB}} \leftarrow\!\!\$\ \mathtt{Comp}(k)$
$\mathtt{Com} \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{RCA},\mathsf{ENC},\mathsf{DEC}}(\mathtt{Dec}_{\mathtt{WB}})$
if $\underset{m \leftarrow\!\!\$\ \mathcal{M}}{Pr}[\mathtt{Decomp}(\mathtt{Com})(\mathtt{Enc}(k,m)) = m] \geq \delta$
$\wedge\ |\mathtt{Com}| \leq |\mathtt{Dec}_{\mathtt{WB}}| - \lambda$
  return 1
else return 0

$\mathsf{RCA}()$

$\mathtt{Dec}'_{\mathtt{WB}} \leftarrow\!\!\$\ \mathtt{Comp}(k)$
return $\mathtt{Dec}'_{\mathtt{WB}}$

$\mathsf{ENC}(m)$

$c \leftarrow\!\!\$\ \mathtt{Enc}(k,m)$
return c

$\mathsf{DEC}(c)$

$m \leftarrow \mathtt{Dec}(k,c)$
return m

4 Constructions of White-Box Cryptography

In this section, we first discuss existing white-box constructions and then present our own construction with a security reduction for (δ,λ)-incompressibility, assuming one-way permutations.

4.1 Existing constructions

The white-box implementations of standardized cryptographic primitives that have been published in [10,29,9,8,40,26] unfortunately turned out insecure with respect to key extraction (see e.g. [7,36]). In turn, more recent works [11,5,6] follow different approaches to construct white-box implementations for alternative (non-standardized) primitives. In [11, Sec. 6], DLPR build a white-box encryption scheme based on a public-key encryption scheme which is secure under their security notions of one-wayness under chosen plaintext attacks and incompressibility. Their implementation is based on the RSA cryptosystem [35]. They first consider the RSA cryptosystem as a symmetric cipher and then use the asymmetric properties of RSA to prove the white-box properties. Likewise, SWP [37] show that public-key encryption systems can first be interpreted as a symmetric encryption algorithm, so that one can then use the asymmetric properties to argue about IND-CPA and IND-CCA security.

Bogdanov and Isobe [5] propose a family of *white-box secure block ciphers* called SPACE, and Bogdanov, Isobe and Tischhauser [6] present an improvement of these designs called SPNbox. The authors claim that these designs are secure under their models for *weak* and *strong space hardness*, a variant of the DLPR model for incompressibility. Their designs are notable in that they present the first symmetric-style construction for an incompressible white-box encryption scheme. The security of their design is based on symmetric cryptanalysis techniques. In turn, a recent construction by FKKM [14] comes with a security reduction. The reduction reduces incompressibility to a novel symmetric-style assumption. Our construction below will improve upon FKKM by moving to the (symmetric) standard-assumption of one-way permutations. Another difference

between FKKM and our construction is that FKKM restricted the adversary to return bits of the key rather than arbitrary strings. Such a restriction, potentially, could enable expansion via secret-sharing, which is highly compressible when allowing for arbitrary compression algorithms. We remove this restriction.

4.2 Incompressible constructions for white-box encryption

In this subsection, we provide an incompressible white-box encryption scheme and an incompressible white-box decryption scheme. We start by introducing our main tool, namely a pseudorandom function that admits a computationally (δ, λ)-incompressible implementation. Then we show that if a PRF admits a computationally (δ, λ)-incompressible implementation, then there is a $(\delta, \lambda - \mathrm{o}(1))$-incompressible white-box encryption scheme and a $(\delta, \lambda - \mathrm{o}(1))$-incompressible decryption scheme. Finally, we construct a computationally incompressible PRF, assuming one-way permutations. Jumping ahead, we note that our incompressible PRF construction makes use of a length-doubling, doubly half-injective pseudorandom generator, a new tool that we introduce and construct in this work, based on one-way permutations.

Computationally incompressible pseudorandom functions. In the following, we consider PRFs whose message and key length are identical, unless stated explicitly otherwise.

Definition 7 (PRF-implementation). *Let f be a PRF. We call a pair of deterministic polynomial-time algorithms $(F, \mathtt{Comp}_{\mathtt{PRF}})$ an* implementation *of the PRF f with expansion α if the following hold:*

Key expansion $\forall k \in \{0,1\}^* \ |K| = \alpha \cdot |k|$, *where* $K = \mathtt{Comp}_{\mathtt{PRF}}(k)$.
Functionality-preservation $\forall k \in \{0,1\}^* \ \forall x \in \{0,1\}^{|k|} f(k,x) = F(K,x)$, *where* $K = \mathtt{Comp}_{\mathtt{PRF}}(k)$.

Definition 8 (computational PRF-incompressibility). *An implementation $(F, \mathtt{Comp}_{\mathtt{PRF}})$ of a PRF f with expansion factor α is called computationally (δ, λ)-incompressible, if the following hold:*

Pseudorandomness $\mathtt{Comp}_{\mathtt{PRF}}(U_n)$ *is computationally indistinguishable from* $U_{\alpha n}$.
Incompressibility *For any PPT computable leakage function* $\mathtt{Leak}$ *and any PPT computable adversary $\mathcal{S}$, it holds that, if* $|\mathtt{Leak}(U_{\alpha n})| \leq \alpha n - \lambda$, *then the probability that the experiment* $\$\text{-}\mathit{PRF}\text{-}\mathit{INC}^{\mathtt{Leak},\mathcal{S}}$ *returns* 1 *is less than* δ.

$\underline{\$\text{-}\mathit{PRF}\text{-}\mathit{INC}^{\mathtt{Leak},\mathcal{S}}}$

$K \leftarrow\!\!\$\ U_{\alpha n}$
$\mathtt{aux} \leftarrow\!\!\$\ \mathtt{Leak}(K)$
$x \leftarrow\!\!\$\ \{0,1\}^n$
$y \leftarrow\!\!\$\ \mathcal{S}(\mathtt{aux}, x)$
return $(y \stackrel{?}{=} F(K,x))$

$\underline{\mathit{PRF}\text{-}\mathit{INC}^{\mathtt{Leak},\mathcal{S}}}$

$k \leftarrow\!\!\$\ \{0,1\}^n$
$K \leftarrow \mathtt{Comp}_{\mathtt{PRF}}(k)$
$\mathtt{aux} \leftarrow\!\!\$\ \mathtt{Leak}(K)$
$x \leftarrow\!\!\$\ \{0,1\}^n$
$y \leftarrow\!\!\$\ \mathcal{S}(\mathtt{aux}, x)$
return $(y \stackrel{?}{=} F(K,x))$

In the $\$\text{-PRF-INC}^{\mathtt{Leak},\mathcal{S}}$ game, the key K is not generated via $\mathtt{Comp}_{\mathtt{PRF}}$, but sampled randomly from the distribution $U_{\alpha n}$. The leakage function $\mathtt{Leak}$ outputs several bits of information of K, which are saved in $\mathtt{aux}$. The adversary $\mathcal{S}$ tries to compute the value y by using $\mathtt{aux}$ instead of the complete key K. The following claim states that due to the pseudorandomness of the key, the success probability of the adversary in the PRF incompressibility game $\$\text{-PRF-INC}^{\mathtt{Leak},\mathcal{S}}$ does not depend (except for a negligible amount) on whether the game uses a real key or a random key. The statement follows directly from the pseudorandomness property of $(F, \mathtt{Comp}_{\mathtt{PRF}})$.

Claim 1. *Let f be a PRF. If $(F, \mathtt{Comp}_{\mathtt{PRF}})$ is a (δ, λ)-incompressible implementation of the PRF f, then for any PPT computable leakage function $\mathtt{Leak}$ and any PPT computable adversary $\mathcal{S}$, it holds that, if $|\mathtt{Leak}(U_{\alpha n})| \leq \alpha n - \lambda$, then the probability that the experiment $\text{PRF-INC}^{\mathtt{Leak},\mathcal{S}}$ returns 1 is at most negligibly greater than δ.*

An incompressible white-box encryption scheme. We now use an incompressible PRF to construct an incompressible white-box encryption scheme. Hereby, we focus on integrity features, i.e., the hardness of producing valid ciphertexts from a compressed algorithm. We achieve this via a message authentication code (MAC) which is generated using the large key K. Additionally, our construction achieves confidentiality via an authenticated encryption scheme which makes use of a small key k'' for encrypting the plaintext and MAC. Since the key k'' is very short in comparison to K, it does not affect the incompressibility of our scheme significantly. An authenticated encryption scheme is a symmetric encryption scheme that satisfies ciphertext integrity and indistinguishability under chosen plaintext attacks. For simplicity, in the following, we assume an authenticated encryption scheme whose key generation algorithm $\mathtt{AKgen}$ samples uniformly random keys of the same length as the security parameter.

Construction 1 (incompressible white-box encryption scheme)**.** *Let* $(\mathtt{AKgen}, \mathtt{AEnc}, \mathtt{ADec})$ *be an authenticated encryption scheme. Let f be a PRF and let $(F, \mathtt{Comp}_{\mathtt{PRF}})$ be an implementation of f with expansion factor α. We construct $\mathsf{WBEnc} = (\mathtt{Kgen}, \mathtt{Enc}, \mathtt{Dec}, \mathtt{Comp})$ as given in Figure 1.*

Theorem 1 (Incompressibility). *If PRF f admits a computationally (δ, λ)-incompressible implementation F, then white-box encryption scheme WBEnc in Construction 1 is a $(\delta, \lambda - n - o(1))$-incompressible white-box encryption scheme.*

Proof. Given a pair of adversaries $(\mathcal{A}, \mathtt{Decomp})$ against (δ, λ)-incompressibility, we need to construct a pair of adversaries $(\mathtt{Leak}, \mathcal{S})$ against the $(\delta, \lambda - n - \mathrm{o}(1))$-incompressibility of the PRF implementation F. The adversary $\mathtt{Leak}$ receives as input the key K, then draws a key k'', builds $\mathtt{Enc}_{\mathtt{WB}}$ as $C[K, k'']$ and runs $\mathcal{A}$ on $\mathtt{Enc}_{\mathtt{WB}}$. The adversary $\mathtt{Leak}$ then emulates the oracles that $\mathcal{A}$ expects as follows: $\mathtt{Comp}$ is a deterministic algorithm and thus, the recompilation algorithm would always return the same program $\mathtt{Enc}_{\mathtt{WB}}$ to $\mathcal{A}$ and so does $\mathtt{Leak}$. Likewise, $\mathtt{Enc}_{\mathtt{WB}}(\cdot)$ and $\mathtt{Enc}(k, \cdot)$ are functionally equivalent, and thus, $\mathtt{Leak}$ can perfectly emulate

$\mathtt{Kgen}(1^n)$	$\mathtt{Enc}(k, m)$	$\mathtt{Dec}(k, c)$
$k' \leftarrow\!\!\$\ \{0,1\}^n$	$k' \leftarrow k[0 : n-1]$	$k' \leftarrow k[0 : n-1]$
$k'' \leftarrow\!\!\$\ \{0,1\}^n$	$k'' \leftarrow k[n : 2n-1]$	$k'' \leftarrow k[n : 2n-1]$
$k \leftarrow k'\|\|k''$	$t \leftarrow f(k', m)$	$\tau \leftarrow \mathtt{ADec}(k'', c)$
return k	$\tau \leftarrow (m, t)$	$(m, t) \leftarrow \tau$
	$c \leftarrow\!\!\$\ \mathtt{AEnc}(k'', \tau)$	**if** $t = f(k', m)$ **return** m.
	return c	**else return** $\perp$

$\mathtt{Comp}(k)$	$C[K, k''](m)$
$k' \leftarrow k[0 : n-1]$	$t \leftarrow F(K, m)$
$k'' \leftarrow k[n : 2n-1]$	$\tau \leftarrow (m, t)$
$K := \mathtt{Comp}_{\mathtt{PRF}}(k')$	$c \leftarrow\!\!\$\ \mathtt{AEnc}(k'', \tau)$
$\mathtt{Enc}_{\mathtt{WB}} := C[K, k''](.)$	**return** c
return $\mathtt{Enc}_{\mathtt{WB}}$	

Fig. 1. Construction: Incompressible white-box encryption scheme based on PRF f and an authenticated encryption scheme.

$\mathtt{Enc}(k, \cdot)$ by running $\mathtt{Enc}_{\mathtt{WB}}(\cdot)$. Finally, to emulate the decryption oracle, the adversary $\mathtt{Leak}$ computes a function that is functionally equivalent to $\mathtt{Dec}(k, \cdot)$ as follows: On input a ciphertext (m, t), the adversary $\mathtt{Leak}$ first decrypts using k'' and then re-computes the PRF on the message m, using K, and checks whether the value is equal to t. If yes, $\mathtt{Leak}$ returns m. Else, $\mathtt{Leak}$ returns $\perp$ to the adversary. Eventually, $\mathcal{A}$ produces some output $\mathtt{Com}$ that $\mathtt{Leak}$ outputs together with k'', i.e., $\mathtt{aux} := (\mathtt{Com}, k'')$.

Finally, we need to construct the adversary $\mathcal{S}$ from the algorithm $\mathtt{Decomp}$. Given the leakage $\mathtt{aux}$ and a value x, the adversary $\mathcal{S}$ runs $\mathtt{Decomp}$ on $\mathtt{aux}$ and obtains a ciphertext c that is an encryption of a pair (x, t) under k''. $\mathcal{S}$ decrypts c using k'' and returns t.

Analysis. Note that $\mathtt{Enc}_{\mathtt{WB}}$, encoded as a Turing machine, is a constant number of bits larger than K and thus, a compressing adversary can strip off those additional bits needed for the Turing machine encoding whence the loss of a constant in λ. By the winning condition of (δ, λ)-incompressibility, $\mathcal{S}$ returns the correct PRF value if and only if $\mathtt{Decomp}(\mathtt{Com})$ returns a ciphertext that decrypts to the correct message. Thus, if $(\mathcal{A}, \mathtt{Decomp})$ satisfies the winning condition with probability greater than δ, so does $(\mathtt{Leak}, \mathcal{S})$. □

In the next subsection, we present a white-box decryption scheme based on an incompressible PRF. Afterwards, in Section 5, we construct an incompressible PRF.

4.3 An incompressible white-box decryption scheme.

For constructing a white-box decryption scheme we focus on the hardness of recovering the message from the ciphertext. Note that analogous to our encryption scheme presented in Construction 1, our decryption scheme can be augmented by adding an authenticated encryption scheme with a comparatively short key on top of it and thus upgrade it to a full authenticated decryption scheme.

Construction 2 (incompressible white-box decryption scheme)**.** *Let f be a PRF and let $(F, \mathtt{Comp}_{\mathtt{PRF}})$ be an implementation of f with expansion factor α. We construct $\mathsf{WBDec} = (\mathtt{Kgen}, \mathtt{Enc}, \mathtt{Dec}, \mathtt{Comp})$ as given in Figure 2.*

$\mathtt{Kgen}(1^n)$	$\mathtt{Enc}(k, m)$	$\mathtt{Dec}(k, c)$	$\mathtt{Comp}(k)$	$C[K](c)$
$k \leftarrow_{\$} \{0,1\}^*$	$r \leftarrow_{\$} \{0,1\}^{\lvert k \rvert}$	$(r, p) \leftarrow c$	$K := \mathtt{Comp}_{\mathtt{PRF}}(k)$	$(r, p) \leftarrow c$
return k	$\mathsf{pad} \leftarrow f(k, r)$	$\mathsf{pad} \leftarrow f(k, r)$	$\mathtt{Dec}_{\mathtt{WB}} := C[K](.)$	$\mathsf{pad} \leftarrow F(K, r)$
	$p \leftarrow m \oplus \mathsf{pad}$	$m \leftarrow p \oplus \mathsf{pad}$	**return** $\mathtt{Dec}_{\mathtt{WB}}$	$m \leftarrow p \oplus \mathsf{pad}$
	$c \leftarrow (r, p)$	**return** m		**return** m
	return c			

Fig. 2. Construction of an incompressible white-box decryption scheme based on a PRF f.

Theorem 2 (Incompressibility). *If a PRF f admits a computationally (δ, λ)-incompressible implementation F, then the white-box decryption scheme WBDec in Construction 2 is a $(\delta, \lambda - o(1))$-incompressible white-box decryption scheme.*

The proof is analogous to the proof of Theorem 1 and thus omitted.

5 Incompressible PRFs from OWPs

The main theorem that we will prove in this section is the following.

Theorem 3. *Assume that one-way permutations exist. Let α be a function in the security parameter n such that for all n, $\alpha(n) > n$ and such that for all n, $\alpha(n)$ is a power of 2. Then, there exists a PRF with a (δ, λ)-incompressible implementation with $\delta = 1 - \frac{\lambda_n}{\alpha} + \mathsf{negl}(n)$, where λ_n is the largest integer such that $n \cdot \lambda_n \leq \lambda$.*

We now construct the incompressible PRF that instantiates this theorem. The writing style of this section is aimed at the parts of the cryptographic community that are familiar with the reduction-based approach to cryptography, see e.g., Goldreich's textbooks on the foundations of cryptography for an excellent introduction [17,18]. Recall that we want to construct a PRF that has its

standard small key as well as a much larger, pseudorandom key that cannot be compressed. Towards this goal, we consider the PRF construction by Goldreich, Goldwasser and Micali (GGM [19]). Recall that the GGM idea is to iterate a PRG within a tree structure, where the paths within the tree is determined by the bits of the PRF input x. That is, let g be a length-doubling PRG and let g_0 be its left half and g_1 be its right half. If k is the PRF key, then the GGM PRF is computed as follows:

$$\texttt{GGM}(k, x) := g_{x[|x|]} g_{x[|x|-1]} \circ ... \circ g_{x[3]} \circ g_{x[2]} \circ g_{x[1]}(k)$$

We now provide an incompressible implementation of the GGM PRF.

Construction 3. *The expansion factor of this incompressible implementation of the GGM PRF is $\alpha = 2^\ell$. For $0 \leq j \leq 2^\ell - 1$, the notation $< j >$ refers to the ℓ-bit string that encodes j in binary.*

$f(k, x)$	$\texttt{Comp}_{\texttt{PRF}}(k)$	$F(K, x)$
$y \leftarrow \texttt{GGM}(k, x)$	**for** j **from** 0 **to** $2^\ell - 1$	$(x[1...\ell], x[\ell + 1...\|x\|]) \leftarrow x$
return y	$k_j := \texttt{GGM}(k, < j >)$	$j \leftarrow x[1...\ell]$
	$K \leftarrow k_0 \|\| ... \|\| k_{2^\ell - 1}$	$y \leftarrow \texttt{GGM}(k_j, x[\ell + 1...\|x\|])$
	return K	**return** y

Fig. 3. Construction of an incompressible implementation of the GGM PRF.

For Construction 3, the key expansion property is clear, and the pseudorandomness property follows from the PRF property of the GGM construction. We thus focus on showing incompressibility properties of Construction 3. To do so, intuitively, one needs to argue that if one loses one bit of the key k_j, then one loses one bit of information about *all* PRF values that are located in the corresponding branch of the GGM PRF (which corresponds to evaluations of messages that start by $< j >$. Unfortunately, such a tight connection might not hold generally. Imagine, e.g., the case, that the PRG in the GGM construction ignores one half of its input and only expands the other half of the input hugely. Likewise, it might be the case that certain bits of the input only affect the left half of the output or the right part of the input. To avoid both of those bad properties, we will consider a PRG that is both, left-half injective and right-half injective. We call such a PRG a doubly half-injective pseudorandom generator (DPRG).

Definition 9 (Doubly Half-Injective Pseudorandom Generator). *A* doubly half-injective *pseudorandom generator (DPRG) is a deterministic polynomial-time computable map $g : \{0,1\}^* \to \{0,1\}^*$ such that the following three properties are satisfied:*

Length-doubling *For all $x \in \{0,1\}^*$, it holds that $|g(x)| = 2|x|$. We write $g_0(x)$ for the left half of g and $g_1(x)$ for the right half of g.*
Doubly half-injective *The functions g_0 and g_1 are injective.*
Pseudorandomness *$g(U_n)$ is computationally indistinguishable from U_{2n}.*

Remark. Note that, as g_0 and g_1 are length-preserving, injectivity is equivalent to bijectivity, but we choose the term injectivity because we only need injectivity in our proofs and because one could define analogous properties also for functions with more stretch. For a further discussion of modification of this definition, see the end of this section.

We build on an observation by Garg, Pandey, Srinivasan and Zhandry [15,16] who show that the standard-construction of a PRG from a one-way permutation is left-half-injective and then transform any left-half injective PRG into a doubly half-injective PRG.

Definition 10 (Left-Half-Injective Pseudorandom Generator). *A* left-half-injective *pseudorandom generator is a deterministic polynomially-time computable map $g : \{0,1\}^* \to \{0,1\}^*$ such that the following three properties are satisfied:*

Length-doubling *For all $x \in \{0,1\}^*$, it holds that $|g(x)| = 2|x|$. We write $g_0(x)$ for the left half of g and $g_1(x)$ for the right half of g.*
Half-injective *The function g_0 is injective.*
Pseudorandomness *$g(U_n)$ is computationally indistinguishable from U_{2n}.*

For completeness, we include the proof of left-half-injectivity by Garg, Pandey, Srinivasan and Zhandry [15,16].

Claim 2 ([15,16])**.** *Assuming the existence of one-way permutations, there exist left-half injective, length-doubling PRGs.*

Proof. Let $f' : \{0,1\}^* \to \{0,1\}^*$ be a one-way permutation. Then the Goldreich-Levin hardcore bit [20] implies that there exists a one-way permutation $f : \{0,1\}^* \to \{0,1\}^*$ with hardcore bit $B : \{0,1\}^* \to \{0,1\}$. We define the function $G : \{0,1\}^* \to \{0,1\}^*$, as $G(x) := f^{|x|}(x)||B(x)||B(f(x))||...||B(f^{|x|-1}(x))$. Indeed, $|G(x)| = 2|x|$. The pseudorandomness of G follows from the security of the hardcore bit, see [17], and the left-injectivity follows, as f is a permutation and therefore, for all ℓ, f^ℓ is a permutation, too. □

We can now prove the existence of doubly half-injective pseudorandom generators, based on one-way permutations.

Lemma 1 (Doubly Half-Injective Pseudorandom Generators). *Assuming the existence of one-way permutations, there exist DPRGs.*

The proof follows directly by combining Claim 2 and the following claim.

Claim 3. *If $G = G_0||G_1$ is a left-half injective, length-doubling PRG, where G_0 denotes its left, injective half, then g is doubly half-injective PRG, where g is defined as*

$$g(x_0||x_1) := G_0(x_0)||G_1(x_0) \oplus G_0(x_1)||G_0(x_1)||G_1(x_1) \oplus G_0(x_0),$$

where $||$ denotes concatenation and where $\oplus$ binds stronger than $||$ and where w.l.o.g., we consider even length $|x|$ and denote x_0 the left half of x and x_1 the right half of x.

Proof. We need to show that g is a doubly half-injective PRG, i.e., we need to prove (1) that each half of g is injective and (2) that the output of g is pseudorandom.

Double Half-Injectivity. We show that $g_0(x_0||x_1) = G_0(x_0)||G_1(x_0) \oplus G_0(x_1)$ is injective. The injectivity of g_1 then follows analogously. Let $w_0||w_1$ be such that $g_0(w_0||w_1) = g_0(x_0||x_1)$. Firstly note that G_0 is a permutation and therefore, $x_0 = w_0$. Plugging this equality into $G_1(w_0) \oplus G_0(w_1) = G_1(x_0) \oplus G_0(x_1)$, we obtain that $G_0(w_1) = G_0(x_1)$. As G_0 is a permutation, it follows that $w_1 = x_1$.

Pseudorandomness. We now prove the pseudorandomness property. We denote by U_n^0, U_n^{00}, U_n^{01}, U_n^1, U_n^{10}, U_n^{11} independent, uniform distributions on n bits. We use that the output of the PRG $G_0(U_n^0)||G_1(U_n^0)$ is computationally indistinguishable from $U_n^{00}||U_n^{01}$ and that $G_0(U_n^1)||G_1(U_n^1)$ is computationally indistinguishable from $U_n^{10}||U_n^{11}$. We get

$$\begin{aligned}
& G_0(U_n^0)||G_1(U_n^0) \oplus G_0(U_n^1)|| && G_0(U_n^1)||G_1(U_n^1) \oplus G_0(U_n^0) \\
\overset{c}{\approx}\ & U_n^{00}||U_n^{01} \oplus G_0(U_n^1)|| && G_0(U_n^1)||G_1(U_n^1) \oplus U_n^{00} \\
\overset{c}{\approx}\ & U_n^{00}||U_n^{01} \oplus U_n^{10}|| && U_n^{10}||U_n^{11} \oplus U_n^{00} \\
\overset{s}{\approx}\ & U_n^{00}||U_n^{01}|| && U_n^{10}||U_n^{11}
\end{aligned}$$

The last step follows, as U_n^{01} and U_n^{11} are independent from the other uniform distributions. We thus proved that G is a pseudorandom generator. Note that the restriction on even input length can be removed by using G_0 and G_1 with matching input and output length (G_1 needs to output strings that are one bit longer than those output by G_0.) and by truncating the output of G_1 appropriately when creating the padding for the shorter half. This concludes the proof of Claim 3. □

We now prove the incompressibility properties of the GGM pseudorandom function when based on a DPRG.

Claim 4. *Let f be the GGM PRF using a DPRG $g = g_0||g_1$. We denote by m the input length of the input x to the PRF. Then for each pair of randomized, possibly inefficient algorithms $(\mathtt{Leak}, \mathcal{S})$, there exists a randomized possibly inefficient algorithm $\mathcal{P}$ such that the probability that the following two experiments return 1 is equal.*

$\$\text{-}\mathit{PRF}\text{-}\mathit{INC}^{\mathtt{Leak},\mathcal{S}}$	$\$\text{-}\mathit{KEY}\text{-}\mathit{INC}^{\mathtt{Leak},\mathcal{P}}$
$k \leftarrow\!\!\$\, U_n$	$k \leftarrow\!\!\$\, U_n$
$\mathtt{aux} \leftarrow\!\!\$\, \mathtt{Leak}(k)$	$\mathtt{aux} \leftarrow\!\!\$\, \mathtt{Leak}(k)$
$x \leftarrow\!\!\$\, \{0,1\}^m$	
$y \leftarrow\!\!\$\, \mathcal{S}(\mathtt{aux}, x)$	$k' \leftarrow\!\!\$\, \mathcal{P}(\mathtt{aux})$
return $(y \stackrel{?}{=} f(k,x))$	**return** $(k' \stackrel{?}{=} k)$

Moreover, for each pair of possibly inefficient algorithms $(\mathtt{Leak}, \mathcal{P})$*, there exists a randomized possibly inefficient algorithm* $\mathcal{S}$ *such that the probability that the two experiments* $\$\text{-}\mathit{PRF}\text{-}\mathit{INC}^{\mathtt{Leak},\mathcal{S}}$ *and* $\$\text{-}\mathit{KEY}\text{-}\mathit{INC}^{\mathtt{Leak},\mathcal{P}}$ *return* 1 *is equal.*

Proof. We observe that for each $x \in \{0,1\}^m$, the function $f(\cdot, x)$ is a permutation as, depending on the bits of x, it applies the functions g_0 and g_1 several times subsequently to the input k. As g_0 and g_1 are permutations, we have a fixed sequence of permutations (depending on the bits of x) that we apply to k. A fixed sequence of permutations is a permutation as well. Therefore, any unpredictability on k immediately translates into unpredictability on the function values of the PRF. We now prove this statement formally. We use the notation $f_x(\cdot)$ for $f(\cdot, x)$ to emphasize that x is fixed and now, for each pair of algorithms $(\mathtt{Leak}, \mathcal{S})$, construct and algorithm $\mathcal{P}$ (left column). We also describe, how for each pair of algorithms $(\mathtt{Leak}, \mathcal{P})$, one can construct an algorithm $\mathcal{S}$ (right column).

$\mathcal{P}(\mathtt{aux})$	$\mathcal{P}(\mathtt{aux}, x)$
$x \leftarrow\!\!\$\, U_n$	
$y \leftarrow\!\!\$\, \mathcal{S}(\mathtt{aux}, x)$	$k' \leftarrow\!\!\$\, \mathcal{P}(\mathtt{aux})$
$k' := f_x^{-1}(y)$	$y := f(k', x)$
return k'	**return** y

As f_x is a permutation, $k' = k$ if and only if $f(k', x) = f_x(k') = f_x(k) = f(k, x)$ and the claim follows. □

In other words, the average min-entropy (see Dodis et al. [12] and Reyzin [34]) of $f(U_n, U_m)$, conditioned on $\mathtt{Leak}(U_n)$, is equal to the average min-entropy of U_n, conditioned on $\mathtt{Leak}(U_n)$. We recall the definition of average min-entropy.

Definition 11 (Average Min-Entropy). *Let* (Y, Z) *be a pair of random variables. The average min-entropy of* Y *conditioned on* Z *is denoted* $\widetilde{H}_\infty(Y|Z)$ *and defined as*

$$-\log \mathbb{E}_{z \leftarrow\!\!\$\, Z}\left[\max_y \Pr[Y = y | Z = z]\right] = -\log\left(\mathbb{E}_{z \leftarrow\!\!\$\, Z}\left[2^{-H_\infty(Y|Z=z)}\right]\right),$$

where $H_\infty(Y|Z=z) = -\log(\max_y \Pr[Y = y|Z = z])$ *denotes min-entropy.*

We can now rephrase Claim 4 as

$$\widetilde{H}_\infty(f(U_n, U_m)|\mathtt{Leak}(U_n)) = \widetilde{H}_\infty(U_n|\mathtt{Leak}(U_n)). \tag{5}$$

Now, we can state the following lemma which concludes the proof of Theorem 3.

Lemma 2. *Let α be a function in the security parameter n such that for all n, $\alpha(n) > n$ and such that for all n, $\alpha(n)$ is a power of 2. Construction 3 is a (δ, λ)-incompressible PRF implementation with expansion factor α of the GGM PRF with $\delta = 1 - \frac{\lambda_n}{\alpha} - \mathsf{negl}(n)$, where λ_n is the largest integer such that $n \cdot \lambda_n \leq \lambda$.*

Proof. We need to show that for each pair of efficient algorithms $(\mathtt{Leak}, \mathcal{S})$, the probability that $\$\text{-}\mathsf{PRF}\text{-}\mathsf{INC}^{\mathtt{Leak},\mathcal{S}}$ returns 1 is smaller than $\delta + \mathsf{negl}(n)$. We will show that this statement even holds for pairs of inefficient algorithms $(\mathtt{Leak}, \mathcal{S})$. That is, the property holds statistically and we need to show that

$$\widetilde{H}_\infty(F(U_{\alpha n}, U_n)|\mathtt{Leak}(U_{\alpha n})) \geq -\log(\delta + \mathsf{negl}(n)). \tag{6}$$

First, remark that as the length of the output of $\mathtt{Leak}$ is upper bounded by λ, we have that

$$\lambda \leq \widetilde{H}_\infty(U_{\alpha n}|\mathtt{Leak}(U_{\alpha n})).$$

We can now split $U_{\alpha n}$ into α blocks of n bits each, where we denote the ith block as $U_{\alpha n}[i]$, and we obtain

$$\widetilde{H}_\infty(U_{\alpha n}|\mathtt{Leak}(U_{\alpha n})) \leq \sum_{i=0}^{\alpha-1} \widetilde{H}_\infty(U_{\alpha n}[i]|\mathtt{Leak}(U_{\alpha n})).$$

We denote by h_i the entropy of the conditional uniform distribution $\widetilde{H}_\infty(U_{\alpha n}[i]|\mathtt{Leak}(U_{\alpha n}))$, which, by Equation 5, is equal to the entropy of the conditional PRF distribution $\widetilde{H}_\infty(f(U_{\alpha n}[i], U_m)|\mathtt{Leak}(U_{\alpha n}))$. Putting all together, we obtain that

$$\lambda \leq \sum_{i=0}^{\alpha-1} h_i, \text{ where} \tag{7}$$

$$\forall 0 \leq i \leq \alpha - 1 : \ 0 \leq h_i \leq n. \tag{8}$$

Recall that we want to show Inequality 6. Using the notation h_i, we can re-phrase Inequality 6 equivalently as

$$S(h_0, ..., h_{\alpha-1}) := \frac{1}{\alpha} \sum_{i=0}^{\alpha-1} 2^{-h_i} \leq \delta + \mathsf{negl}(n)\,. \tag{9}$$

To summarize, we need to find h_0,...,$h_{\alpha-1}$ such that Inequality 7 and Inequality 8 are satisfied and such that the term $S(h_0, ..., h_{\alpha-1})$ on the left-hand side of Inequality 9 is maximized. On the α-dimensional domain that satisfies Inequality 8, the term $S(h_0, ..., h_{\alpha-1})$ is maximized when $h_0 = ... = h_{\alpha-1} = 0$. Moreover, S is

anti-monotone. That is, if $(h'_0, ..., h'_{\alpha-1}) \leq (h_0, ..., h_{\alpha-1})$ component-wise, then $S(h'_0, ..., h'_{\alpha-1}) \geq S(h_0, ..., h_{\alpha-1})$. Moreover, given any point $(h_0, ..., h_{\alpha-1})$ in the domain $[0, n]^\alpha$, the descent of S is least steep in the direction of the largest entry h_i. As S is symmetric, we obtain that under the constraints of Inequality 7 and Inequality 8, S is maximized at $\overline{h} = (n, ..., n, \lambda_{\text{rem}}, 0, ..., 0)$, which contains λ_n entries n and where λ_{rem} is such that $\lambda = \lambda_n \cdot n + \lambda_{\text{rem}}$. We obtain

$$S(\overline{h}) = \frac{1}{\alpha}(\lambda_n \cdot 2^{-n} + 2^{-\lambda_{\text{rem}}} + (\alpha - \lambda_n - 1)) \leq 1 - \frac{\lambda_n}{\alpha} + \mathsf{negl}(n)\,,$$

which concludes the proof of the lemma. □

Discussion on stretch and assumptions.

Note that one can obtain DPRGs with more stretch from a DPRG that is length-doubling simply by first applying the original DPRG and then applying an injective PRG to the left half and an injective PRG to the right half of the output of the DPRG. Also note that a DPRG with stretch 2 implies (is actually equivalent to) the existence of one-way permutations and that one-way permutations imply injective PRGs via the Goldreich-Levin hardcore bit construction [20].

Our construction would also work with a DPRG that stretches its input by more than a factor of 2. Such a function might be constructed based on one-way functions only, as g_0 and g_1 would not be bijective anymore and thus, such a DPRG does not seem to imply one-way permutations unlike a DPRG whose stretch is exactly 2. In the rest of the paper, we considered DPRGs whose stretch exactly 2. We made no attempt to construct DPRGs based on one-way functions only, as one-way permutations are a standard symmetric-type MiniCrypt assumption.[5]

Acknowledgements. The authors would like to thank the anonymous reviewers of CT-RSA 2019 for their useful feedback and, in particular, for helping us to clarify related work. Part of this work was done while Chris Brzuska and Estuardo Alpirez Bock were at the Hamburg University of Technology and they are grateful to NXP for supporting their chair for IT Security Analysis during that period.

References

1. B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, Aug. 2001.

[5] That is, one-way permutations are not known to imply trapdoor functions, and, by the seminal paper of Impagliazzo and Rudich [23], it seems unlikely that anyone would show such an implication anytime soon. See also Impagliazzo [22] for an excellent survey on cryptographic assumptions.

2. M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In T. Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 531–545. Springer, Heidelberg, Dec. 2000.
3. O. Billet, H. Gilbert, and C. Ech-Chatbi. Cryptanalysis of a white box AES implementation. In H. Handschuh and A. Hasan, editors, *SAC 2004*, volume 3357 of *LNCS*, pages 227–240. Springer, Heidelberg, Aug. 2004.
4. A. Biryukov, C. Bouillaguet, and D. Khovratovich. Cryptographic schemes based on the ASASA structure: Black-box, white-box, and public-key (extended abstract). In P. Sarkar and T. Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 63–84. Springer, Heidelberg, Dec. 2014.
5. A. Bogdanov and T. Isobe. White-box cryptography revisited: Space-hard ciphers. In I. Ray, N. Li, and C. Kruegel:, editors, *ACM CCS 15*, pages 1058–1069. ACM Press, Oct. 2015.
6. A. Bogdanov, T. Isobe, and E. Tischhauser. Towards practical whitebox cryptography: Optimizing efficiency and space hardness. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 126–158. Springer, Heidelberg, Dec. 2016.
7. J. W. Bos, C. Hubain, W. Michiels, and P. Teuwen. Differential computation analysis: Hiding your white-box designs is not enough. In B. Gierlichs and A. Y. Poschmann, editors, *CHES 2016*, volume 9813 of *LNCS*, pages 215–236. Springer, Heidelberg, Aug. 2016.
8. J. Bringer, H. Chabanne, and E. Dottax. White box cryptography: Another attempt. Cryptology ePrint Archive, Report 2006/468, 2006. `http://eprint.iacr.org/2006/468`.
9. S. Chow, P. A. Eisen, H. Johnson, and P. C. van Oorschot. White-box cryptography and an AES implementation. In K. Nyberg and H. M. Heys, editors, *SAC 2002*, volume 2595 of *LNCS*, pages 250–270. Springer, Heidelberg, Aug. 2003.
10. S. Chow, P. A. Eisen, H. Johnson, and P. C. van Oorschot. A white-box DES implementation for DRM applications. In J. Feigenbaum, editor, *Security and Privacy in Digital Rights Management, ACM CCS-9 Workshop, DRM 2002*, volume 2696 of *LNCS*, pages 1–15. Springer, 2003.
11. C. Delerablée, T. Lepoint, P. Paillier, and M. Rivain. White-box security notions for symmetric encryption schemes. In T. Lange, K. Lauter, and P. Lisonek, editors, *SAC 2013*, volume 8282 of *LNCS*, pages 247–264. Springer, Heidelberg, Aug. 2014.
12. Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 523–540. Springer, Heidelberg, May 2004.
13. ECRYPT. Ches 2017 capture the flag challenge - the whibox contest, 2017. `https://whibox.cr.yp.to/`.
14. P.-A. Fouque, P. Karpman, P. Kirchner, and B. Minaud. Efficient and provable white-box primitives. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 159–188. Springer, Heidelberg, Dec. 2016.
15. S. Garg, O. Pandey, and A. Srinivasan. Revisiting the cryptographic hardness of finding a nash equilibrium. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 579–604. Springer, Heidelberg, Aug. 2016.
16. S. Garg, O. Pandey, A. Srinivasan, and M. Zhandry. Breaking the sub-exponential barrier in obfustopia. In J. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 156–181. Springer, Heidelberg, May 2017.

17. O. Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001.
18. O. Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.
19. O. Goldreich, S. Goldwasser, and S. Micali. On the cryptographic applications of random functions. In G. R. Blakley and D. Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 276–288. Springer, Heidelberg, Aug. 1984.
20. O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *21st ACM STOC*, pages 25–32. ACM Press, May 1989.
21. L. Goubin, J.-M. Masereel, and M. Quisquater. Cryptanalysis of white box DES implementations. In C. M. Adams, A. Miri, and M. J. Wiener, editors, *SAC 2007*, volume 4876 of *LNCS*, pages 278–295. Springer, Heidelberg, Aug. 2007.
22. R. Impagliazzo. A personal view of average-case complexity. In *Proceedings of the Tenth Annual Structure in Complexity Theory Conference, Minneapolis, Minnesota, USA, June 19-22, 1995*, pages 134–147. IEEE Computer Society, 1995.
23. R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *21st ACM STOC*, pages 44–61. ACM Press, May 1989.
24. R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In S. Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 8–26. Springer, Heidelberg, Aug. 1990.
25. M. Jacob, D. Boneh, and E. W. Felten. Attacking an obfuscated cipher by injecting faults. In J. Feigenbaum, editor, *Security and Privacy in Digital Rights Management, ACM CCS-9 Workshop, DRM 2002, Washington, DC, USA, November 18, 2002, Revised Papers*, volume 2696 of *LNCS*, pages 16–31. Springer, 2003.
26. M. Karroumi. Protecting white-box AES with dual ciphers. In K. H. Rhee and D. Nyang, editors, *ICISC 10*, volume 6829 of *LNCS*, pages 278–291. Springer, Heidelberg, Dec. 2011.
27. H. Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is SSL?). In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 310–331. Springer, Heidelberg, Aug. 2001.
28. T. Lepoint, M. Rivain, Y. D. Mulder, P. Roelse, and B. Preneel. Two attacks on a white-box AES implementation. In T. Lange, K. Lauter, and P. Lisonek, editors, *SAC 2013*, volume 8282 of *LNCS*, pages 265–285. Springer, Heidelberg, Aug. 2014.
29. H. E. Link and W. D. Neumann. Clarifying obfuscation: Improving the security of white-box encoding. Cryptology ePrint Archive, Report 2004/025, 2004. `http://eprint.iacr.org/2004/025`.
30. Mastercard. Mastercard mobile payment sdk, 2017. `https://developer.mastercard.com/media/32/b3/b6a8b4134e50bfe53590c128085e/mastercard-mobile-payment-sdk-security-guide-v2.0.pdf`.
31. E. Alpirez Bock, C. Brzuska, W. Michiels, and A. Treff. On the ineffectiveness of internal encodings - revisiting the dca attack on white-box cryptography. Cryptology ePrint Archive, Report 2018/301, 2018. `https://eprint.iacr.org/2018/301.pdf`.
32. Y. D. Mulder, P. Roelse, and B. Preneel. Cryptanalysis of the Xiao-Lai white-box AES implementation. In L. R. Knudsen and H. Wu, editors, *SAC 2012*, volume 7707 of *LNCS*, pages 34–49. Springer, Heidelberg, Aug. 2013.
33. Y. D. Mulder, B. Wyseur, and B. Preneel. Cryptanalysis of a perturbated white-box AES implementation. In G. Gong and K. C. Gupta, editors, *INDOCRYPT 2010*, volume 6498 of *LNCS*, pages 292–310. Springer, Heidelberg, Dec. 2010.

34. L. Reyzin. Some notions of entropy for cryptography - (invited talk). In S. Fehr, editor, *ICITS 11*, volume 6673 of *LNCS*, pages 138–142. Springer, Heidelberg, May 2011.
35. R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signature and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, 1978.
36. E. Sanfelix, J. de Haas, and C. Mune. Unboxing the white-box: Practical attacks against obfuscated ciphers. Presentation at BlackHat Europe 2015, 2015. `https://www.blackhat.com/eu-15/briefings.html`.
37. A. Saxena, B. Wyseur, and B. Preneel. Towards security notions for white-box cryptography. In P. Samarati, M. Yung, F. Martinelli, and C. A. Ardagna, editors, *ISC 2009*, volume 5735 of *LNCS*, pages 49–58. Springer, Heidelberg, Sept. 2009.
38. Smart Card Alliance Mobile and NFC Council. Host card emulation 101. white paper, 2014. `http://www.smartcardalliance.org/downloads/HCE-101-WP-FINAL-081114-clean.pdf`.
39. B. Wyseur, W. Michiels, P. Gorissen, and B. Preneel. Cryptanalysis of white-box DES implementations with arbitrary external encodings. In C. M. Adams, A. Miri, and M. J. Wiener, editors, *SAC 2007*, volume 4876 of *LNCS*, pages 264–277. Springer, Heidelberg, Aug. 2007.
40. Y. Xiao and X. Lai. A secure implementation of white-box AES. In *2009 2nd International Conference on Computer Science and its Applications*, pages 1–6. IEEE Computer Society, 2009.