1

Introduction to White-Box Cryptography

Pierre GALISSANT and Louis GOUBIN
Laboratoire de Mathématiques de Versailles, UVSQ, CNRS,
Université Paris-Saclay, France

1.1. Introductory remarks

In 1883, Auguste Kerckhoffs' article, *La cryptographie militaire*, was published in the *Journal des sciences militaires*, in which he stated six design rules for military ciphers. Here they are, translated from French:

1) The system must be practically, if not mathematically, indecipherable.

2) It should not require secrecy, and it should not be a problem if it falls into enemy hands.

3) It must be possible to communicate and remember the key without using written notes, and correspondents must be able to change or modify it at will.

4) It must be applicable to telegraph communications.

5) It must be portable and should not require several persons to handle or operate.

6) Lastly, given the circumstances in which it is to be used, the system must be easy to use and should not be stressful to use or require its users to know and comply with a long list of rules.

The second rule, now known as Kerckhoffs's principle, has been completely taken into account in the modern cryptography concepts. More precisely, it is usually

Embedded Cryptography 3,
coordinated by Emmanuel PROUFF, Guénaël RENAULT,
Matthieu RIVAIN and Colin O'FLYNN.

assumed, in most security models, that the attacker has a complete knowledge of the formal specifications of the cryptographic algorithms that are used to secure the communications or the storage of sensitive data.

On top of that, to make security possible, we must assume that the legitimate users (traditionally called Alice and Bob in the two-party scenarios) know a secret data which gives them an advantage over a potential adversary (usually called Charlie). Since this secret *key* has to be involved in the cryptographic computations, and must still remain hidden from the attacker, the classical security model in cryptography makes the assumption that it is not only the key but also the explicit implementation of the algorithm (which includes the key) which cannot be accessed by the attacker. This is the *black-box* model.

Nevertheless, cryptographic algorithms are increasingly deployed in various applications embedded on connected devices, such as smartphones and tablets. In this environment, the capabilities of the adversary can be greatly enhanced, and we should consider an adversary who can access the binary code, modify its execution, tamper with the memory and use existing reverse engineering tools such as debuggers to recover the hidden secrets. This *white-box* model is of course more demanding, and white-box cryptography aims at providing the same security guarantees as in the black-box model, despite this huge new advantage given to the adversary.

In the following sections, we will be able to provide precise security notions that capture this idea of cryptographic security in a white-box model. However, we give here some preliminary remarks to highlight the subtleties which can arise when trying to formalize white-box cryptography.

As a first tentative goal, let us consider the problem of providing an implementation of a block cipher algorithm E_K for which the secret key K is computationally difficult to extract from the given implementation.

If no further constraints are given, it is not difficult to build such a block cipher algorithm, together with a white-box implementation. Indeed, we can choose E defined by:

$$E_K = E'_{h(K)}$$

where E' is a standard block cipher (for instance, AES) and h is a one-way function (for instance, h can be built with a standard hash function). It is obviously easy to build an implementation such that $h(K)$ is easy to recover, but not K. This shows that the difficulty of extracting the key does not completely capture the intuitive notion we have for white-box cryptography.

Another natural idea follows from this first remark: Can we make it computationally difficult, from the implementation of E_K (i.e. the code of the function $x \mapsto E_K(x)$), to find a decomposition of the form:

$$E_K(x) = F(x, G(K)),$$

for which we can explicitly obtain the code of F and the code of G? Note that in this framework, G can contain several elements. For instance, block ciphers typically use a key schedule mechanism such as $G(K) = (K_1, \ldots, K_r)$, with *subkeys* K_i $(1 \leq i \leq r)$, so that we have:

$$E_K(x) = F(x, K_1, ..., K_r)$$

A typical example that would not satisfy this definition is a "one-way" key schedule:

$$K_i = h(K||i) \quad (1 \leq i \leq r),$$

where h is a one-way function (that can easily be built from a standard hash function).

In summary, it seems a good definition of white-box cryptography should forbid the following situation: "the code is independent of K, except constants which depend on K in a deterministic way".

At first sight, we could fear all cryptographic implementations would therefore be forbidden! Fortunately, the possibility remains that G has, among its inputs, an external value r:

$$E_K(x) = F(x, G(K, r))$$

This construction is reminiscent of the randomness used in side-channel countermeasures and historically appeared to be a key idea in the seminal proposals of white-box DES and AES implementations (Chow et al. 2002), where:

$$G(K, r) = \text{tables that depend on } K \text{ and } r.$$

Note that this also opens the way to a useful application: *traitor tracing*. If someone reveals $G(K, r)$, we can recover r (in the extreme case, by exhaustive search on r). This relies on the assumption that it is difficult to deduce $G(K, r')$ from $G(K, r)$ (and a fortiori K from $G(K, r)$).

We can also remark that such a construction can be done in the context of the RSA primitive. The fundamental idea is the following: if the encryption function is built upon $x \mapsto y = x^e \bmod n$, the decryption function is based on $y \mapsto x = y^d \bmod n$ (where d is the inverse of e modulo $\varphi(n)$) and can be implemented as $y \mapsto x = y^{d'} \bmod n$, where d' is an arbitrary large integer such that $d' \equiv d \bmod \varphi(n)$, namely, $d' = d + r\varphi(n)$. For this construction to make sense, we have to assume that e is also a secret exponent (if not, a well-known argument can be used to deduce the factorization of n from d'), so that what we obtain is an example of white-box symmetric algorithm:

$$\begin{cases} K = (e, d, p, q) \\ E_K(y) = y^{d'} \bmod n \\ E_K^{-1}(x) = x^e \bmod n \end{cases}$$

Here, which instructions are executed (or not) may depend on K (typically in the case of *square and multiply*: "if $d'_i = 1$ then $x := x \times y \bmod n$"). However, the knowledge of these executed instructions is equivalent to the knowledge of n and d', which does not allow us to recover $K = (e, d, p, q)$.

1.2. Basic notions for white-box cryptography

The basic security requirement for a white-box implementation is to resist key extraction. However, we should expect more from white-box cryptography and consider various security properties for the white-box implementations; and hopefully we provide formal definitions and security notions for white-box cryptography. In particular, Delerablée et al. (2014) defined some concrete white-box security notions for symmetric encryption schemes. For example, *unbreakability*, *one-wayness*, *incompressibility* and *traceability* are derived from folklore intuitions behind white-box cryptography.

1.2.1. *Unbreakability*

The notion of unbreakability is a very intuitive security notion for white-box cryptography and has been studied since the seminal paper of Chow et al. (2002).

Let us describe the game for unbreakability of a white-box compiler $\mathcal{C}_\mathcal{S}$ corresponding to a given cryptographic algorithm $\mathcal{S}$:

– draw at random key k in private keyspace $K_\mathcal{S}$;

– the adversary $\mathcal{A}$ gets the program $\mathcal{C}_\mathcal{S}(k)$ from the compiler;

– the adversary $\mathcal{A}$ returns a key guess $\hat{k}$ in time, and τ knows $\mathcal{C}_\mathcal{S}(k)$;

– the adversary $\mathcal{A}$ succeeds if $k = \hat{k}$.

DEFINITION 1.1.– Let $\mathcal{S}$ be a cryptographic algorithm, $\mathcal{C}_{\mathcal{S}}$ a white-box compiler for this algorithm $\mathcal{S}$ and let $\mathcal{A}$ be any adversary. We define the probability of the adversary $\mathcal{A}$ to succeed in the unbreakability game by:

$$Succ_{\mathcal{A},\mathcal{C}_{\mathcal{S}}} := \mathbb{P}[k \leftarrow K; \mathcal{P} = \mathcal{C}_{\mathcal{S}}(k), \mathcal{A}(\mathcal{P}) = \hat{k}; k = \hat{k}]$$

We say that $\mathcal{C}_{\mathcal{S}}$ is (τ,ϵ)-unbreakable if for any adversary $\mathcal{A}$ running in time τ, $Succ_{\mathcal{A},\mathcal{C}_{\mathcal{S}}} \leq \epsilon$.

1.2.2. *Incompressibility*

The notion of incompressibility is a stronger security notion, first formally defined by Delerablée et al. (2014), where the study of this notion is motivated as a software countermeasure against code-lifting attacks.

We now describe, for any $\sigma > 0$, the game for incompressibility for a white-box compiler $\mathcal{C}_{\mathcal{S}}$ corresponding to a given cryptographic algorithm $\mathcal{S}$:

– draw at random a key k in private keyspace $K_{\mathcal{S}}$;

– the adversary $\mathcal{A}$ gets the program $\mathcal{C}_{\mathcal{S}}(k)$ from the compiler;

– the adversary $\mathcal{A}$ returns a program $\mathcal{P}$ knowing $\mathcal{C}_{\mathcal{S}}(k)$;

– the adversary $\mathcal{A}$ succeeds if $\mathcal{P} \approx \mathcal{C}_{\mathcal{S}}(k)$ and $size(\mathcal{P}) \leq \sigma$.

DEFINITION 1.2.– Let $\mathcal{S}$ be a cryptographic algorithm, $\mathcal{C}_{\mathcal{S}}$ a white-box compiler for this algorithm $\mathcal{S}$ and let $\mathcal{A}$ be any adversary. We define the probability of the adversary $\mathcal{A}$ to succeed in the σ-incompressibility game by:

$$Succ_{\mathcal{A},\mathcal{C}_{\mathcal{S}}} := \mathbb{P}[k \leftarrow K; \mathcal{P} = \mathcal{A}(\mathcal{C}_{\mathcal{S}}(k)); \mathcal{P} \approx \mathcal{C}_{\mathcal{S}}(k); \ (size(\mathcal{P}) \leq \sigma)]$$

Moreover, we say that $\mathcal{C}_{\mathcal{S}}$ is (σ,τ,ϵ)-incompressible if for any adversary $\mathcal{A}$, Time($\mathcal{A}$)+Time($\mathcal{P}$) $< \tau$ implies $Succ_{\mathcal{A},\mathcal{C}_{\mathcal{S}}} \leq \epsilon$.

REMARK 1.1.– In a more general case, the definition can include a parameter δ that allows the program $\mathcal{P}$ to agree with the targeted function with probability δ.

REMARK 1.2.– Note that if a compiler is incompressible, then it is unbreakable for reasonable security levels: the key-recovery is indeed an extreme compression of a white-box implementation.

The definition of incompressibility we state here is a slightly modified version from the initial definition of Delerablée et al. (2014). Indeed, the latter does not constrain the running time of the program $\mathcal{P}$, which leaves the possibility of a trivial

way of compressing any white-box algorithm, namely, by using brute force: an attacker can compute a few (plaintext, ciphertext) pairs and code the brute-force attack on the primitives that are white-boxed, and then code the computation of the primitive, including the obtained key. This program can be made with few lines of code and is functionally equivalent to the initial white-box implementation, but has an unreasonable running time. The definition above makes use of a new time constraint: the sum of the running time of the attacker and the produced program must be less than a constant τ representing the whole computation time allowed.

REMARK 1.3.– The previous modification does not necessarily invalidate all the proofs found in the literature for incompressibility. Indeed, in some of these proofs, the programs produced by the attacker are restrained by a model (for example, Ideal Group Model), whereas the program we mentioned here does not fit into this kind of model.

1.2.3. *One-wayness*

Let us describe the game for one-wayness of a white-box compiler $\mathcal{C}_\mathcal{S}$ corresponding to a given symmetric encryption algorithm $\mathcal{S}$:

– draw at random a key k in private keyspace $K_\mathcal{S}$;

– the adversary $\mathcal{A}$ gets the program $\mathcal{C}_\mathcal{S}(k)$ from the compiler;

– the adversary $\mathcal{A}$ gets the ciphertext c corresponding to a randomly selected plaintext m;

– the adversary $\mathcal{A}$ returns a guess $\hat{m}$ in time τ knowing $\mathcal{C}_\mathcal{S}(k)$;

– the adversary $\mathcal{A}$ succeeds if $m = \hat{m}$.

DEFINITION 1.3.– Let $\mathcal{S}$ be a symmetric encryption algorithm, $\mathcal{C}_\mathcal{S}$ a white-box compiler for this algorithm $\mathcal{S}$ and let $\mathcal{A}$ be any adversary. We define the probability of the adversary $\mathcal{A}$ to succeed in the one-wayness game by:

$$Succ_{\mathcal{A},\mathcal{C}_\mathcal{S}} := \mathbb{P}[k \leftarrow K, m \leftarrow M; \mathcal{P} = \mathcal{C}_\mathcal{S}(k), c = \mathcal{S}(m), \mathcal{A}(\mathcal{P}, c) = \hat{m}; m = \hat{m}]$$

We say that $\mathcal{C}_\mathcal{S}$ is (τ,ϵ)-one-way if for any adversary $\mathcal{A}$ running in time τ, $Succ_{\mathcal{A},\mathcal{C}_\mathcal{S}} \leq \epsilon$.

REMARK 1.4.– This one-wayness has the following consequence: the cryptographic algorithm can be used as a public-key cryptosystem. It is indeed possible to see the obtained white-box implementation as a public key, and the key K of the algorithm as the private key. The one-wayness property implies that the public key can only be used to encrypt messages, whereas the decryption requires the knowledge of the private key. This illustrates the power of this security notion.

1.3. Proposed (and broken) solutions

1.3.1. *Block ciphers*

Many attempts have been made to construct white-box implementations for standard block ciphers in recent years.

DES obfuscation methods were first proposed by Chow et al. (2002) at the DRM 2002 workshop. The simplest method ("naked DES") was cryptanalyzed by Chow et al. (2003) themselves at the SAC 2002 conference; an improved method was also cryptanalyzed by Jacob et al. (2002) and by Link and Neuman (2004); the strongest known method ("nonstandard-DES") was also cryptanalyzed independently by Wyseur et al. (2007) and by Goubin et al. (2007) at the SAC 2007 conference.

A specific obfuscation method for AES was proposed by Chow et al. (2002) at SAC 2002, and later cryptanalyzed by Billet et al. (2004) at SAC 2004 (see also Billet (2005); PhD Thesis, defended in December 2005).

Further construction attempts followed, for instance, by Link and Neumann (2004), Bringer et al. (2006), Xiao and Lai (2009), and by Karroumi (2010), but they were also shown to be insecure sooner or later, by De Mulder et al. (2010) at INDOCRYPT, De Mulder et al. (2012) at SAC, Lepoint et al. (2013) at SAC, De Mulder et al. (2013), and Lepoint and Rivain (2013). The WhibOx 2017 and 2019 contests showed that even with hidden designs, producing unbreakable and one-way AES implementations in pure software is a difficult open problem.

This illustrates that reaching the *unbreakability* property – and a fortiori the *incompressibility* property – are already difficult when implementing standard block ciphers such as 3DES or AES.

Concerning the *one-wayness* security notion, we already mentioned that it allows us to tranform a (symmetric) block cipher into a public-key cryptosystem, which gives a first hint that it is probably even more difficult to obtain than unbreakability.

More precisely, for usual block ciphers, the one-wayness problem appears to have a close relationship with the problem of "functional decomposition". Standard block ciphers are indeed built from several rounds (for instance, 16 rounds for DES or 10 rounds for AES), which leads to typical implementations as a loop corresponding to a functional decomposition:

$$E_K = f_r \circ \ldots \circ f_1$$

Hence, inverting E_K boils down to inverting each f_i, which is likely to be an easy task.

REMARK 1.5.– A natural idea would be to compute functional compositions of the type $f_i \circ f_j$, such that the "functional decomposition" is computationally hard. However, the problem for classical block ciphers is that the algebraic degree of such compositions grows quickly. This can even be seen as an unavoidable consequence of the fundamental principle stated by Shannon (1949) paper: breaking a "good" cipher should require "as much work as solving a system of simultaneous equations in a large number of unknowns of a complex type". Therefore, the composition cannot be done via the representation as polynomial systems, and new strategies seem to be required here.

REMARK 1.6.– White-box constructions (with the unbreakability and one-wayness properties) are possible if we are allowed to choose an (ad hoc) block cipher. This is reminiscent of public-key cryptography and can easily be illustrated in the context of multivariate cryptography. In a nutshell, a multivariate algorithm makes use of a function A that can be represented as a system of n multivariate polynomials in n variables and can be easily inverted. In the asymmetric setting, the secret key comprises two secret invertible linear (of affine) functions s and t, and the corresponding public key is obtained as $t \circ A \circ s$, one of the security assumptions being that recovering A from this public key in computationally difficult (this is usually called the "Isomorphism of polynomials with two secrets" problem).

This construction can be converted into a symmetric block cipher:

$$\begin{cases} K = (s, t, A) \\ E_K(y) = (t \circ A \circ s)(y), \text{where } t \circ A \circ s \text{ is given as a system of polynomials} \\ E_K^{-1}(x) = s^{-1} \circ A^{-1} \circ t^{-1}(x) \end{cases}$$

In summary, the idea here is that multivariate cryptography can make K difficult to extract from the implementation of E_K, without having a huge algebraic degree for the "global" (and public) description of E_K.

1.3.2. *Asymmetric algorithms*

While many candidates have been publicly proposed to construct white-box implementations of block ciphers, almost no white-box implementations for public-key algorithms have been published up to now, in spite of numerous research efforts.

In their works, Feng et al. (2020) and Zhang et al. (2020), claim to achieve unbreakability for asymmetric cryptosystems. However, their proposals require a

non-standard verification process, which makes them irrelevant for genuine public-key applications.

To the best of our knowledge, the first candidate that does not change the underlying scheme is due to Barthelemy (2020a, 2020b), who proposed a white-box implementation of a scheme suggested by Aguilar Melchior et al. (2016) whose (black-box) security is based on the computational difficulty of the RLWE (ring learning with errors) problem over the cyclotomic ring $R_q = \mathbb{Z}/q\mathbb{Z}[X]/(X^n + 1)$.

Lucas Barthelemy's implementation of the decryption algorithm makes use of the NTT transformation and RNS representations to reduce the computation to small look-up tables, which can in turn be transformed using ideas dating back to the SAC 2002 seminal paper of Chow et al. (2020), together with additive of multiplicative masking based on homomorphic properties of the cryptographic scheme. However, a fatal flaw was found (and acknowledged by Lucas Barthelemy in his PhD thesis): the core part of the white-box implementation consists of trying to prevent the key extraction for a function of the form $\alpha_2 - \alpha_1 \cdot sk$, where (α_1, α_2) is the ciphertext we want to decrypt, and sk is the secret key. This function is linear in α_1 and α_2, and this linear dependence on the elements coming from the ciphertext remains true, even after applying the NTT transformations and using the representations RNS. It is therefore very easy for an attacker to find the coefficients of these linear transformations (which depend on sk), then sk itself.

The WhibOx 2021 contest showed that for the ECDSA algorithm, even with hidden design, getting an unbreakable implementation is out of reach: all the implementations proposed were quickly broken. In-depth analyses of the generic attacks and problems these implementations suffered were provided by Barbu et al. (2022) and Bauer et al. (2022).

Galissant and Goubin (2022) proposed a concrete white-box implementation for the well-known hidden field equations (HFE) signature algorithm (a signature algorithm belonging to the multivariate family of public key algorithms) for a specific set of internal polynomials, providing the first white-box implementation of a public key algorithm, together with an extensive security analysis providing strong arguments for both unbreakability and incompressibility. For a security level 2^{80}, the public key size is approximately 62.5 MB and the white-box implementation of the signature algorithm has a size of approximately 256 GB.

This is a promising research direction and some variants are currently being investigated to improve the size of the white-box implementation and adapt it to various security levels.

1.4. Generic strategies to build white-box implementations

1.4.1. *DCA and countermeasures*

In the white-box model, the attacker has access all the details of the implementation of a (known) cryptographic algorithm, including a given secret key, and the goal of the attacker is to recover the secret key. Up to now, most candidate implementations (of DES, AES, substitution linear-transformation ciphers, etc.) have been broken. Most of the time, the attacks use various cryptanalytic techniques, which require knowledge and understanding of all the implementation principles and details.

In the past years, *generic attacks* have emerged that apply to white-box implementations, irrespective of their (secret) designs and which consist of translating usual hardware attacks to the white-box setting. In particular, Sanfelix et al. (2015), Bos et al. (2016) at CHES, described differential computational analysis (DCA), whose principle is to apply side-channel analysis (SCA) techniques to so-called *computational traces* composed of all the intermediate results of the computation (bus transfers, register allocations, memory addresses, etc.). More precisely, a dynamic binary instrumentation (DBI) framework can be used to build software traces and then mount an analogue of differential power attack (DPA) on these software traces. The advantage of this technique is that – as DPA in the case of smart card implementations – the attacker does not need to know (and to analyze) the very details of the implementation. The attack can be launched in an automated way on candidate white-box implementations.

For instance, Bos et al. (2016) describe several examples illustrating the power of this DCA: the authors show that their method can break many challenge implementations, which utilize many of the ideas used up to now to build white-box implementations. This shows that an already stated principle (namely, a whitebox implementation must in particular resist all kinds of side-channel attacks) had probably not been considered seriously enough.

The lack of random source in a white-box implementation (at run-time) is a reason for the weakness against DCA, but theoretically this does not rule out the possibility of resistant white-box implementations. Therefore, a first challenge is to provide a theoretical explanation of the fact that white-box implementations, based on look-up tables, can be attacked by SCA, even when they are “hidden” by randomly chosen bijections (e.g. in DES and AES implementation by Chow et al. (2002)).

A second challenge consists of elaborating specific countermeasures against this new kind of attack.

REMARK 1.7.– As noted by Jacob et al. (2002), and Sanfelix et al. (2015), differential fault analysis (DFA) can also be directly applied to the white-box setting, so that resisting these attacks is also a challenge for future research.

1.4.2. *Using fully homomorphic encryption (FHE)*

When considering DCA, we have to limit the power of the attacker, usually by bounding the "order" of such attacks. Classical ways of protecting the key are making use of the *secret sharing* principle, leading to so-called *masking* countermeasures.

In the white-box setting, such a limit is less relevant. We cannot expect noise to make the complexity of the attack grow exponentially with the DCA order. Is it therefore natural to push DCA to its limits, and try to obtain "infinite order" countermeasures? The intuition can be viewed as follows. DCA-type masking countermeasures of order n are based on the idea that the attacker cannot control more than n values, so that computing the algorithm with a "multiparty computation"-like implementation can prevent the attack. In the context of delegated computations, when no limit is assumed about the number of parties controlled by the attacker, multiparty computation is not sufficient any more, and has to be replaced by FHE.

For the more general problem of obfuscation, methods based on hard computational problems have indeed been derived from fully homomorphic encryption and the universal oblivious Turing machine. Pippenger and Fischer (1979) proved that a two-tape oblivious Turing machine can simulate any non-oblivious Turing machine with only logarithmic slowdown. The idea is then to homomorphically run the universal oblivious Turing machine, with two inputs:

$$Prog' = FHE.Encrypt(Prog),$$

where $Prog$ is the program to be computed in an obfuscated way. Of course, the program does not appear in the form $Prog$ but only in the form $Prog'$, pre-computed during the creation of the obfuscated software.

$$x' = FHE.Encrypt(x),$$

where x is the input of $Prog$.

To resist partial evaluation attacks and mixed input attacks, as noticed by Garg et al. (2013), the final decryption of the result has to be conditional. The condition is twofold:

– Check the proof of computation of the universal oblivious Turing machine that testifies that the program $Prog'$ was indeed run (in an FHE way) on the input x', and also that $x' = FHE.Encrypt(x)$ was correctly computed.

– Verify a digital signature of $Prog'$, so as to authentify the executed program.

This idea can directly be adapted to the white-box context by using FHE to encrypt only K instead of the whole program $Prog$. For the conditional decryption, two possibilities arise. Conditional decryption can be executed within a dedicated tamper-resistant hardware, as illustrated by Bitansky et al. (2011) and Döttling et al. (2011). One more challenging direction consists of replacing this hardware part by an obfuscated (software) program. To achieve this, a line of research – starting from a paper by Garg et al. (2013) that develops a complex design based on branching programs and multilinear maps – aims at obtaining generic obfuscation methods (which here would only use the fact that the conditional decryption can be written as a NC^1 circuit). However, they are still highly non-practical.

1.4.3. *White-box solutions with the help of a (small) tamper-resistant hardware*

Alpirez Bock et al. (2020) considered (at ASIACRYPT) an alternative use of such a tamper-resistant hardware. They build a hardware-bound white-box key derivation function (WKDF) on top of a standard (black-box) key derivation function (KDF). In a nutshell, if the adversary uses its hardware access, they are able to evaluate the WKDF, but if they have no access to the relevant hardware values, for example, in case of a code-lifting attack, then they learn nothing about the WKDF values. The design, based on techniques published by Sahai and Waters (2014), requires puncturable pseudorandom functions (PRFs, which are equivalent to one-way functions) and indistinguishability obfuscation (*iO*). Although interesting from a theoretical point of view, the current state of the art of *iO* makes the construction highly impractical.

In comparison, as described in section 1.4.2, white-box resistance can be achieved using FHE together with a (relatively small) tamper-resistant hardware that computes the conditional decryption operation. The performance overhead due to FHE is rather high, but the solution remains realistic in case we really need a single call to the hardware at the end of the computation.

Other ways of using a secure hardware component have been considered in the context of white-box cryptography (seen as a particular case of software obfuscation). For instance, Anderson (2008) described the following idea. The program to be obfuscated can be written on the encrypted memory tape of a Turing machine. Each time an operation has to be executed, the whole tape is sent to the hardware component, which decrypts it, executes the next instruction, reencrypts the

tape and sends it back to the software part. This is of course very time consuming and requires a huge number of exchanges between the software and the hardware token.

A more efficient solution was described by Goyal et al. (2010). By building on techniques from resettably secure computation (due to Goyal and Sahai (2009)), they gave a general positive result for stateless oblivious reactive functionalities under standard cryptographic assumption. This result also provides the first general feasibility result for program obfuscation using stateless tokens. As a side result, they also propose constructions of non-interactive secure computation for general reactive functionalities with stateful tokens, which can be adapted to hardware-aided white-box constructions.

1.5. Applications of white-box cryptography

Real-world applications of white-box cryptography are numerous. Below are some typical examples illustrating potential use cases of white-box cryptography, either in a symmetric model or in a public key setting.

1.5.1. *EMV payments on NFC-enabled smartphones without secure element*

In recent years, the payment industry has shown great interest in the extension of the EMV specifications to mobile transactions via near field communication (NFC). In that scenario, the usual contactless smart card is emulated by an NFC-compliant mobile phone or wearable device such as a smart watch. This is referred to as *host card emulation* (HCE). Unfortunately, however, mobile platforms do not provide access to a secure element to third-party applications: the SIM card belongs to the telecommunication operator and handset manufacturers keep any form of trusted hardware for their own needs. These emerging applications are therefore facing the challenge of being as secure as a tamper-resistant hardware, although being totally based on software. White-box cryptography is currently the only approach to secure these applications and compensate the security risks inherent to common embedded operating systems such as Android. By hard-coding the EMV keys into the application code itself, as suggested in the EMVCo requirements documentation in 2019, white-box cryptography tries to achieve a notion of *tamper-resistant software*. Similarly, Mastercard Cloud-Based Payments (MCBP) is a secure and scalable software-based solution developed to digitize card credentials and enable both contactless and remote payment transactions. In this context, in 2017, Mastercard specifically recommended the use of white-box implementation for the secure storage of payment tokens.

1.5.2. *Software DRM mechanisms for digital contents*

Digital right management (DRM) is a set of techniques whereby subscribers get access to protected content under a number of conditions (access rights). Video on-demand and mobile TV are typical examples of DRM-protected services. Here again, in the absence of a hardware cryptographic module, a white-box implementation of the content decryption algorithm under an individual user key prevents the key from being recovered and re-used by third-parties (piracy based on key sharing and redistribution).

1.5.3. *Mobile contract signing*

The eIDAS regulation (EU Reg. No. 910/2014) came into force on July 1, 2016 in the 28 member states of the EU, and introduced the *end of the smart card dogma*, in the sense that the signing capability can now be implemented by purely software means as long as they fulfill specific requirements through a qualification procedure. Electronic signatures also become legal evidence that cannot be denied by sovereign authorities or in court. By relaxing constraints on the signing utility, the eIDAS regulation opens the way to software-only solutions for digital signatures. As a result, a rapid emergence of mobile contract signing is anticipated in the near future. The user experience is straightforward: a contract (or any form of document in that respect) is downloaded on the mobile device, reviewed by the human user, digitally signed locally and the legally binding signature is returned to a back-end server in the cloud, where it is validated and archived. Now, the need for the signing application to be eIDAS-qualified imposes (depending on the qualification level) resisting security threats pertaining to mobile platforms and most particularly logical attacks where some form of external control is exerted through malware, typically in an attempt to steal the signing key(s) stored on the device. White-box cryptography is the only approach that effectively puts the signing key(s) out of reach of logical attacks on the operating system. Combined with countermeasures against code lifting, white-box cryptography is expected to take a major role in the adoption and deployment of eIDAS-based services in the EU.

1.5.4. *Cryptocurrencies and blockchain technologies*

Most solutions to store cryptocurrencies and perform transactions on the blockchain are today based on a hardware token (USB stick, smart card) or on a mobile application. While the former provide adequate security, it is inconvenient for the wider usage. For the latter case on the other hand, the security often relies on the operating system of the mobile device and the principle of application sandboxing. Given the wide variety of mobile OS versions on the field, strictly relying on the operating system to protect critical assets (such as money) is very hazardous and

should always be avoided. This raises a strong need for the design of security solutions for pure-software cryptocurrency wallet against all kind of threats such as stealing malwares. In order to protect the cryptographic keys intrinsically involved in cryptocurrencies and blockchain technologies, white-box cryptography is essential. As concerns digital signatures, ECDSA is currently the most used algorithm (for instance, Bitcoin and Ethereum), but alternatives are considered, either for other cryptocurrencies or to prepare for the post-quantum era.

1.6. Notes and further references

From a historical perspective, the term "white-box cryptography" was introduced by Chow et al. (2002, 2003) in their seminal papers in relation to the following situation: "When the attacker has internal information about a cryptographic implementation, choice of implementation is the sole remaining line of defense" (Chow et al. 2003).

– Section 1.1. Auguste Kerckhoffs published his seminal paper in Kerckhoffs (1883a). See also Kerckhoffs (1883b), entitled *La cryptographie militaire, ou les chiffres usités en temps de guerre, avec un nouveau procédé de déchiffrement applicable aux systèmes à double clef.*

– Section 1.2. The paper by Delerablée et al. (2014) about white-box definitions was published in 2013 in the procedings of the SAC conference. The seminal paper of Chow et al. (2002) was published in the proceedings of the DRM 2002 workshop; see also their paper at SAC 2002 (Chow et al. 2003).

– Section 1.3. DES obfuscation methods, first proposed by Chow et al. (2002), were cryptanalyzed by Chow et al. (2003), Jacob et al. (2002), Link and Neumann (2004), Goubin et al. (2007) and Wyseur et al. (2007).

The white-box implemention for AES by Chow et al. (2003) was later cryptanalyzed by Billet et al. (2004) (see also Billet (2005)). Other constructions were proposed by Link and Neumann (2004), Bringer et al. (2006), Xiao and Lai (2009) and Karroumi (2011), but were all broken by De Mulder et al. (2010, 2013a, 2013b), Lepoint and Rivain (2013); Lepoint et al. (2014), Lepoint and Rivain (2013); Lepoint et al. (2014). During the WhibOx Organizing Committee (2017) and WhibOx Organizing Committee (2019) contests, all the AES proposals were eventually broken.

In his famous paper "Communication theory of secrecy systems" (Shannon 1949), Claude Shannon discussed cryptography from an information theory point of view, thus providing foundations of modern cryptography.

In the asymmetric context, white-box implementations were claimed by Feng et al. (2020) and Zhang et al. (2020), but they have to modify the verification process, so that the cryptosystem does not fit the original public key framework. Lucas Barthelemy's candidate (Barthelemy 2020b) is a white-box implementation of a public key encryption scheme, a variant of a scheme suggested by Aguilar Melchor et al. (2016). The white-box implementation uses ideas from Chow et al. (2003), but flaws were acknowledged by Barthelemy in his PhD thesis (Barthelemy 2020a): the linear dependences allowing us to recover the secret key can be seen in the equations in section 4.2.

During the WhibOx Organizing Committee (2021) contest, all of the proposed ECDSA white-box implementations were eventually broken. Detailed analyses of the attacks were given by Barbu et al. (2022) and Bauer et al. (2022).

The recent proposal by Galissant and Goubin (2022) is a white-box implementation of a variant of HFE, a signature algorithm belonging to the multivariate family of public key algorithms.

– Section 1.4. The idea of differential computational analysis (DCA) was introduced by Sanfelix et al. (2015) and Bos et al. (2016). Its principle originates in side-channel analysis (SCA) techniques (see, for instance, Kocher et al. (1999) or Brier et al. (2004)), applied to computational traces, instead of power (or electro-magnetic) traces.

The idea of using differential fault analysis (DFA) in the context of white-box implementations was mentioned by Jacob et al. (2002) and Sanfelix et al. (2015). Basic notions about DFA can be found in the well-known papers of Boneh et al. (1997) and Biham and Shamir (1997).

A detailed survey of fully homomorphic encryption can be found in Marcolla et al. (2022). The property that a two-tape oblivious Turing machine can simulate any non-oblivious Turing machine with only logarithmic slowdown is due to Pippenger and Fischer (1979).

The idea of using a conditional decryption operation to resist partial evaluation attacks and mixed input attacks was described by Garg et al. (2013) at FOCS. This operation can in turn be executed either in a tamper-resistant hardware, as illustrated by Bitansky et al. (2011) and Döttling et al. (2011), or in an obfuscated way, following a line of research initiated by Garg et al. (2013).

A construction of a hardware-bound white-box key derivation function (WKDF) was proposed at ASIACRYPT 2020 by Alpirez Bock et al. (2020). It is based on an idea of Sahai and Waters (2014), according to which white-box can be obtained from puncturable pseudorandom functions (PRFs) and indistinguishability obfuscation (iO).

Other constructions using a secure hardware component to obtain obfuscation properties were proposed by Anderson (2008) and at TCC 2010 by Goyal et al. (2010). The latter is based on techniques from resettably secure computation, due to Goyal and Sahai (2009) at EUROCRYPT.

– Section 1.5. The extension of the EMV specifications to mobile transactions via near field communication (NFC) was specified in EMVCo (2008). In 2019, the EMVCo requirements documentation (EMVCo 2019) suggested to hard-code the EMV keys into the application code itself. In the same spirit, Mastercard Cloud-Based Payments (MCBP) (Mastercard 2014) aims at digitizing card credentials, and Mastercard has specifically recommended the use of white-box implementation for the secure storage of payment tokens (Mastercard 2017).

1.7. References

Aguilar Melchor, C., Barrier, J., Guelton, S., Guinet, A., Killijian, M.-O., Lepoint, T. (2016). NFLlib: NTT-based fast lattice library. In *CT-RSA 2016*, Sako, K. (ed.). Springer, Heidelberg.

Alpirez Bock, E., Brzuska, C., Fischlin, M., Janson, C., Michiels, W. (2020). Security reductions for white-box key-storage in mobile payments. In *ASIACRYPT 2020*, Moriai, S. and Wang, H. (eds). Springer, Heidelberg.

Anderson, W.E. (2008). On the secure obfuscation of deterministic finite automata. Cryptology ePrint Archive, Report 2008/184 [Online]. Available at: https://eprint.iacr.org/2008/184.

Barbu, G., Beullens, W., Dottax, E., Giraud, C., Houzelot, A., Li, C., Mahzoun, M., Ranea, A., Xie, J. (2022). ECDSA white-box implementations: Attacks and designs from WhibOx 2021 contest. Report 2022/385, Cryptology ePrint Archive [Online]. Available at: https://eprint.iacr.org/2022/385.

Barthelemy, L. (2020a). A first approach to asymmetric white-box cryptography and a study of permutation polynomials modulo 2^n in obfuscation. PhD Thesis, Sorbonne Université, Paris.

Barthelemy, L. (2020b). Toward an asymmetric white-box proposal. Cryptology ePrint Archive, Report 2020/893 [Online]. Available at: https://eprint.iacr.org/2020/893.

Bauer, S., Drexler, H., Gebhardt, M., Klein, D., Laus, F., Mittmann, J. (2022). Attacks against white-box ECDSA and discussion of countermeasures – A report on the WhibOx contest 2021. Report 2022/448, Cryptology ePrint Archive [Online]. Available at: https://eprint.iacr.org/ 2022/448.

Biham, E. and Shamir, A. (1997). Differential fault analysis of secret key cryptosystems. In *CRYPTO'97*, Kaliski, B.S. Jr. (ed.). Springer, Heidelberg.

Billet, O. (2005). Cryptologie multivariable. PhD Thesis, Université de Versailles-Saint-Quentin-en-Yvelines, Versailles.

Billet, O., Gilbert, H., Ech-Chatbi, C. (2004). Cryptanalysis of a white box AES implementation. In *SAC 2004*, Handschuh, H. and Hasan, A. (eds). Springer, Heidelberg.

Bitansky, N., Canetti, R., Goldwasser, S., Halevi, S., Kalai, Y.T., Rothblum, G.N. (2011). Program obfuscation with leaky hardware. In *ASIACRYPT 2011*, Lee, D.H. and Wang, X. (eds). Springer, Heidelberg.

Boneh, D., DeMillo, R.A., Lipton, R.J. (1997). On the importance of checking cryptographic protocols for faults (extended abstract). In *EUROCRYPT'97*, Fumy, W. (ed.). Springer, Heidelberg.

Bos, J.W., Hubain, C., Michiels, W., Teuwen, P. (2016). Differential computation analysis: Hiding your white-box designs is not enough. In *CHES 2016*, Gierlichs, B. and Poschmann, A.Y. (eds). Springer, Heidelberg.

Brier, E., Clavier, C., Olivier, F. (2004). Correlation power analysis with a leakage model. In *CHES 2004*, Joye, M. and Quisquater, J.-J. (eds). Springer, Heidelberg.

Bringer, J., Chabanne, H., Dottax, E. (2006). White-box cryptography: Another attempt. Cryptology ePrint Archive, Report 2006/468 [Online]. Available at: https://eprint.iacr.org/2006/468.

Chow, S., Eisen, P.A., Johnson, H., van Oorschot, P.C. (2002). A white-box DES implementation for DRM applications. In *Security and Privacy in Digital Rights Management*. Springer, Heidelberg.

Chow, S., Eisen, P.A., Johnson, H., van Oorschot, P.C. (2003). White-box cryptography and an AES implementation. In *SAC 2002*, Nyberg, K. and Heys, H.M. (eds). Springer, Heidelberg.

De Mulder, Y., Wyseur, B., Preneel, B. (2010). Cryptanalysis of a perturbated white-box AES implementation. In *INDOCRYPT 2010*, Gong, G. and Gupta, K.C. (eds). Springer, Heidelberg.

De Mulder, Y., Roelse, P., Preneel, B. (2013a). Cryptanalysis of the Xiao-Lai white-box AES implementation. In *SAC 2012*, Knudsen, L.R. and Wu, H. (eds). Springer, Heidelberg.

De Mulder, Y., Roelse, P., Preneel, B. (2013b). Revisiting the BGE attack on a white-box AES implementation. Cryptology ePrint Archive, Report 2013/450 [Online]. Available at: https://eprint.iacr.org/2013/450.

Delerablée, C., Lepoint, T., Paillier, P., Rivain, M. (2014). White-box security notions for symmetric encryption schemes. In *SAC 2013*, Lange, T., Lauter, K., Lisonek, P. (eds). Springer, Heidelberg.

Döttling, N., Mie, T., Müller-Quade, J., Nilges, T. (2011). Basing obfuscation on simple tamper-proof hardware assumptions. Cryptology ePrint Archive, Report 2011/675 [Online]. Available at: https://eprint.iacr.org/2011/675.

EMVCo (2008). Integrated circuit card specifications for payment systems. Book 2. Security and Key Management. Version 4.2 [Online]. Available at: www.emvco.com.

EMVCo (2019). EMV mobile payment: Software-based mobile payment security requirements. Technical Report [Online]. Available at: https://www.emvco.com/wp-content/uploads/documents/EMVCo-SBMP-16-G01-V1.2SBMPSecurityRequirements.pdf.

Feng, Q., He, D., Wang, H., Kumar, N., Choo, K.R. (2020). White-box implementation of Shamir's identity-based signature scheme. *IEEE Syst. J.*, 14(2), 1820–1829. doi: 10.1109/JSYST.2019.2910934.

Galissant, P. and Goubin, L. (2022). Resisting key-extraction and code-compression: A secure implementation of the HFE signature scheme in the white-box model. Cryptology ePrint Archive, Report 2022/138 [Online]. Available at: https://eprint.iacr.org/2022/138.

Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B. (2013). Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*. IEEE, Berkeley.

Goubin, L., Masereel, J.-M., Quisquater, M. (2007). Cryptanalysis of white box DES implementations. In *SAC 2007*, Adams, C.M., Miri, A., Wiener, M.J. (eds). Springer, Heidelberg.

Goyal, V. and Sahai, A. (2009). Resettably secure computation. In *EUROCRYPT 2009*, Joux, A. (ed.). Springer, Heidelberg.

Goyal, V., Ishai, Y., Sahai, A., Venkatesan, R., Wadia, A. (2010). Founding cryptography on tamper-proof hardware tokens. In *TCC 2010*, Micciancio, D. (ed.). Springer, Heidelberg.

Jacob, M., Boneh, D., Felten, E.W. (2002). Attacking an obfuscated cipher by injecting faults. In *Security and Privacy in Digital Rights Management*. Springer, Heidelberg.

Karroumi, M. (2011). Protecting white-box AES with dual ciphers. In *ICISC 10*, Rhee, K.H. and Nyang, D. (eds). Springer, Heidelberg.

Kerckhoffs, A. (1883a). La cryptographie militaire. *Journal des sciences militaires*, 9, 5–38.

Kerckhoffs, A. (1883b). *La cryptographie militaire, ou les chiffres usités en temps de guerre, avec un nouveau procédé de déchiffrement applicable aux systèmes à double clef*. Librairie Militaire de L. Baudoin, Paris.

Kocher, P.C., Jaffe, J., Jun, B. (1999). Differential power analysis. In *CRYPTO'99*, Wiener, M.J. (ed.). Springer, Heidelberg.

Lepoint, T. and Rivain, M. (2013). Another nail in the coffin of white-box AES implementations. Cryptology ePrint Archive, Report 2013/455 [Online]. Available at: https://eprint.iacr.org/2013/455.

Lepoint, T., Rivain, M., De Mulder, Y., Roelse, P., Preneel, B. (2014). Two attacks on a white-box AES implementation. In *SAC 2013*, Lange, T., Lauter, K., Lisonek, P. (eds). Springer, Heidelberg.

Link, H.E. and Neumann, W.D. (2004). Clarifying obfuscation: Improving the security of white-box encoding. Cryptology ePrint Archive, Report 2004/025 [Online]. Available at: https://eprint.iacr.org/2004/025.

Marcolla, C., Sucasas, V., Manzano, M., Bassoli, R., Fitzek, F.H.P., Aaraj, N. (2022). Survey on fully homomorphic encryption, theory, and applications. *Proceedings of the IEEE*, 110(10), 1572–1609.

Mastercard (2014). Mastercard cloud-based payments – Mobile payment application functional description. Version 1.0.

Mastercard (2017). Mastercard mobile payment SDK security guide for MP SDK v1.0.6. Version 2.0. Technical Report [Online]. Available at: https://developer.mastercard.com/media/32/b3/b6a8b4134e50bfe53590c128085e/mastercard-mobile-payment-sdk-security-guide-v2.0.pdf.

Pippenger, N. and Fischer, M.J. (1979). Relations among complexity measures. *J. ACM*, 26(2), 361–381. doi: 10.1145/322123.322138.

Sahai, A. and Waters, B. (2014). How to use indistinguishability obfuscation: Deniable encryption, and more. In *46th ACM STOC*, Shmoys, D.B. (ed.). ACM Press, New York.

Sanfelix, E., de Haas, J., Mune, C. (2015). Unboxing the white-box: Practical attacks against obfuscated ciphers. Presentation, BlackHat Europe [Online]. Available at: https://www.blackhat.com/eu-15/briefings.html.

Shannon, C.E. (1949). Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4), 656–715.

WhibOx Organizing Committee (2017). CHES 2017 CTF challenge – WhibOx contest [Online]. Available at: https://whibox.io/contests/2017/.

WhibOx Organizing Committee (2019). CHES 2019 CTF challenge – WhibOx contest [Online]. Available at: https://whibox.io/contests/2019/.

WhibOx Organizing Committee (2021). CHES 2021 CTF challenge – WhibOx contest [Online]. Available at: https://whibox.io/contests/2021/.

Wyseur, B., Michiels, W., Gorissen, P., Preneel, B. (2007). Cryptanalysis of white-box DES implementations with arbitrary external encodings. In *SAC 2007*, Adams, C.M., Miri, A., Wiener, M.J. (eds). Springer, Heidelberg.

Xiao, Y. and Lai, X. (2009). A secure implementation of white-box AES. In *2009 2nd International Conference on Computer Science and its Applications*. IEEE, Jeju.

Zhang, Y., He, D., Huang, X., Wang, D., Choo, K.R., Wang, J. (2020). White-box implementation of the identity-based signature scheme in the IEEE P1363 standard for public key cryptography. *IEICE Trans. Inf. Syst.*, 103-D(2), 188–195.