TECHNICAL REPORT

ISO/IEC TR 24485

First edition
2022-10

Information security, cybersecurity and privacy protection — Security techniques — Security properties and best practices for test and evaluation of white box cryptography

Sécurité de l'information, cybersécurité et protection de la vie privée — Techniques de sécurité — Propriétés de sécurité et bonnes pratiques pour les essais et l'évaluation de la cryptographie en boîte blanche

Reference number
ISO/IEC TR 24485:2022(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see https://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *Information security, cybersecurity and privacy protection*.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

The white box cryptography (WBC) is a specific implementation method for cryptographic algorithms where the secret key and the algorithm are entangled, so that the user can freely stimulate the latter. However, the secret key is deeply engraved in the implementation, such that it is hard to extract and is unambiguous. Furthermore, it is also difficult to update the WBC implementation in order to change the key.

WBC is typically used for implementation when secure cryptographic module functionality (such as a tamper-resistance device or a physically unclonable function) is unavailable for the protection of secrets.

Business cases include DRM (digital right management) and HCE (host card emulation), in the contexts of media protection and payment applications. WBC implementations are widely deployed on mobile devices, where the cryptographic implementation is provided over the top.

The purpose of this document is to provide best practices on security assurance and to facilitate users to assess the security level of several WBC implementations. It is important to share common information and best practices about test and evaluation of WBC security.

Information security, cybersecurity and privacy protection — Security techniques — Security properties and best practices for test and evaluation of white box cryptography

1 Scope

This document introduces security properties and provides best practices on the test and evaluation of white box cryptography (WBC). WBC is a cryptographic algorithm specialized for a key or secret, but where the said key cannot be extracted.

The WBC implementation can consist of plain source code for the cryptographic algorithm and/or of a device implementing the algorithm. In both cases, security functions are implemented to deter an attacker from uncovering the key or secret.

Security properties consist in the secrecy of security parameters concealed within the implementation of the white box cryptography. Best practices for the test and evaluation includes mathematical and practical analyses, static and dynamic analyses, non-invasive and invasive analyses.

This document is related to ISO/IEC 19790 which specifies security requirements for cryptographic modules. In those modules, critical security parameters (CSPs) and public security parameters (PSPs) are the assets to protect. WBC is one solution to conceal CSPs inside of the implementation.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 19790, *Information technology — Security techniques — Security requirements for cryptographic modules*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 19790 and the following apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1
white box cryptography
WBC
physically unprotected cryptographic software implementation aimed at mitigating the risk of extracting secrets obfuscated within the implementation

3.2
white-boxing generator
program which dynamically generates a new *white box cryptography* (3.1) instance

3.3
disembedding
extracting software binary code from its intended platform in a malicious way to transfer it to a separate platform amenable to analysis

4 Security properties of white box cryptography

4.1 Implementation of a white box cryptography

4.1.1 General

A white box cryptography (WBC) implementation is a software code implementing a cryptographic function. The implementation can be varied, to match constraints in terms of performance (code size, execution duration, power or energy consumption, etc.), usability (portability, description language, etc.) and of security.

4.1.2 Description of a WBC

A WBC implementation can be available in different forms including:

a) Source code or compiled code (e.g. binary code).

b) Plain or intentionally obfuscated code.

c) Standalone code with an API (application programming interface) to a software cryptographic library (hence with a clearly defined boundary) or code embedded into an application (hence a difficult task to separate the WBC code from the applicative code).

d) For greatest risk mitigation, typically, a unique WBC instance is deployed for each obfuscated cryptographic operation or key usage for each platform instance. It is indeed common to distribute a unique WBC implementation (corresponding to a unique key) per device. Therefore, there is a need for a program to generate WBC implementations. There are various levels of availability of the WBC code generator (as known as white-boxing generator), including:

    1) It is public. This is either because it has been stolen or leaked by an insider, or because it is used in a trustworthy context where it is paramount that the system be open so as to avoid risks of backdoors; or

    2) Only some aspects are known. This means the design rationale can be public, but be itself configured with some secrets; or

    3) It is completely secret.

The generation of the WBC implementation (displayed as “WBC exe”) from the WBC generator is depicted in Figure 1 showing the two contexts of white-boxing generator; when the attacker knows it and when it does not know it. The “grey” situation, where some aspects are known, can also be envisioned. Depending on the case, the analysis strategy can differ. The input/output correspond to the user interface of the crypto-algorithm except the secret key, for example:

a) If the application is a block-cipher input is Plaintext and output is Ciphertext

b) If the application is a digital-signature input is the message to be signed and output is the signature

c) If the application is a Message Authentication Code, input is the message to be authenticated and output is the authentication tag

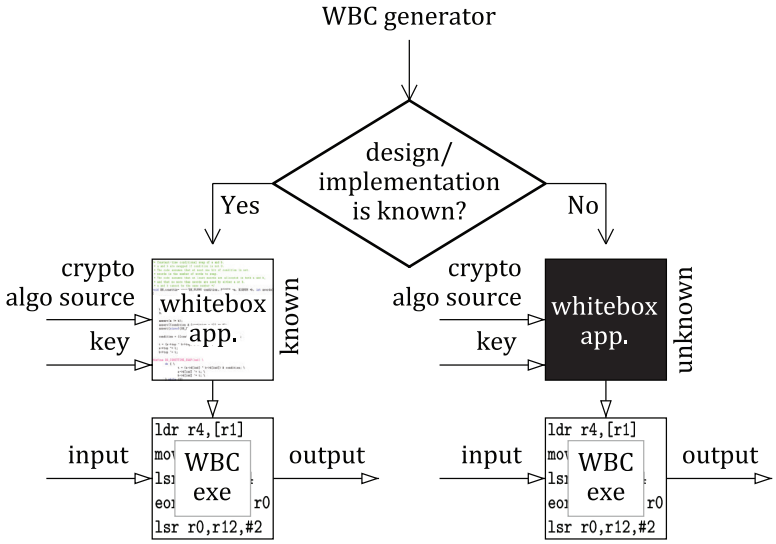

Figure 1 — Context of an evaluation of a white-box implementation

4.1.3 Adherence between WBC code and the device hosting it

The WBC implementation can be specialized to run correctly only on a specific device. Examples of ways to tie a code to a device include the use of embedded keys and/or counters, emulation software or secret virtual machine/sandbox.

This adherence requires some security features. Typically, the device specific values are supposed to be hard to extract by the attackers. Similarly, it is supposed that it is hard for an attacker to extract any secret associated with the WBC operation, or to manipulate the WBC operation in a manner that has an adverse security impact.

4.2 WBC attack path(s)

4.2.1 General

The state-of-the-art attacks techniques are reviewed in 4.2. In practice, an adversary would combine several attack steps to achieve his goal, such as uncovering the secret key concealed in the WBC implementation. The attacker would also typically finish his attack by validating the found key (or key candidates) by predicting other ciphertexts from unseen plaintexts.

4.2.2 De-embedding of code (code lifting)

The WBC implementations are better studied if indeed they can be freely observed and manipulated by an attacker. Therefore, an attacker initially attempts to access the code. This can be achieved when the code is downloaded inside the device. There are many paths via which an attacker possibly can lift the code in theory, but that platforms can mitigate that risk though various controls such as the provision of secure code download channels and various access control measures. But the device can implement some protections. For instance,

a) the device possibly can enforce some restrictions to the way the WBC is called, such as limited number of calls (in absolute or per unit of time);

b) the device possibly can randomize the execution.

Therefore, recovering the code is a means to analyse it with more freedom. Typically, it is possible to have a full access to the code and also to its simulation. Moreover, one can also modify the code. For instance, in order to replace the return value of the function with a constant. In such conditions, the

code can be analysed in a view to identify and then exploit weaknesses in the WBC design. Code lifting can take on two aspects:

a) Overcoming copy protection: the attacker is required to find a method to read the WBC code out of the device, though there can be no functional means to actually extract it.

b) Running the code on a clone platform, which is instrumented to observe and/or alter step-by-step the code while it is executed; this can require building such testing environment, possibly specific to the implementation being investigated.

4.2.3 Device analysis

Some implementations of WBC can make it hard to achieve code lifting. It is therefore possible for an attacker to perform indirect observation and/or perturbation of the code, e.g. by spying on memory usage or other shared resources (such as cache memories, etc.). Alternatively, observation/perturbation can be perpetrated from the outside, by the exploitation of so-called side-channels[1]. Those are unintended physical interactions that a remote attacker can measure from the WBC implementation under analysis.

4.2.4 Code analysis

The WBC code can be analysed by classical reverse-engineering techniques. It is customary to classify them in two categories: static and dynamic analyses. Static analysis represents the code as an AST (abstract syntax tree), and manages to typically simplify it. Dynamic analysis executes the code and observes the traces, which include register values (including the special registers, such as the flags), the memory usage (such as the stack and the heap, etc.). Some mixed techniques exist. Typically, concolic (shortcut for "concrete and symbolic") consists in simulating the code, albeit with formal values for the variables. In this sense, the execution is "symbolic". Nonetheless, because cryptographic codes are complex, and also because WBC implementations further intentionally make it more complex, the symbolic execution becomes too computationally intensive as the analysis steps further deep in the code. It is, thus, possible in concolic frameworks for the tester (the attacker) to fix concrete (numerical) values to some variables (which as otherwise manipulated as formal terms). Dynamic and concolic techniques aim at better understanding the code, with a view to clarifying how the secret key and the (public or not) algorithms can be separated, thereby allowing to extract parts or whole portion of the secret key.

During static or dynamic analysis, values can be analysed visually. However, such technique can be time consuming and it further requires human skills and attention. An alternative is to automate the analysis of the concrete or formal data collected during the analysis, by the use of some distinguishers. Those are statistical tools which allow for checking whether a link exists between intermediate variables and guessed variables (e.g. through a model). When the model is parametrized by a portion of the secret key to recover, the distinguisher acts as an oracle to decide which key portion is the most likely. Such technique based on concrete simulation of lifted code is explained in the DCA (differential computation analysis[2]).

4.3 WBC usages

4.3.1 General

This document focuses on secrets engraved within the WBC implementation. However, without loss of generality, it can be envisioned to use WBC design rationale to store a PSP (public security parameter), such as a public key meant to verify a digital signature, inside the implementation. The WBC implementation takes care not to allow the attacker to substitute his PSP to the genuine PSP. This case is not precluded in this document, but neither further developed. Indeed, algorithms using PSP such as digital signatures typically return a binary value (true if the signature is correct, false otherwise). It is thus straightforward to wrap the WBC code with a code which modifies the returned value according to a forged PSP chosen by the attacker.

Use-cases or white-boxing for standard algorithms classes are further developed in 4.3.2 to 4.3.5.

4.3.2 Symmetric encryption

Secret key algorithms, such as block ciphers (e.g. as specified in ISO/IEC 18033-3, or lightweight block ciphers, as specified in ISO/IEC 29192-2), are natural targets for being white-boxed. The secret is the key. The algorithms can be used in any mode of operation: the secondary security parameters (such as initialization vectors, counters, tweaks, etc.) need not be secret but are expected to remain unaltered, otherwise the mode of operation can be broken. Indeed, the general recommendations of proper usage apply, the cryptography being white-boxed or not.

All higher-level protocols, such as “authenticated encryption”, are also natural applications for WBC.

4.3.3 Asymmetric encryption / signature

Asymmetric protocols usually consist of three algorithms: key generation, public key usage (for encryption or signature verification) and private key usage (such as decryption or signature generation). The private key operation, for RSA and elliptic curve cryptography, are natural targets. For example, the asymmetrical algorithms standardized in ISO/IEC 14888 (all parts) specify “digital signatures with appendix”. They can advantageously be protected by white-boxing.

Post-quantum cryptographic algorithms (such as lattice-based, code-based, multivariate cryptosystems) implement key Encapsulation/Decapsulation Mechanisms and digital signatures, which are also a venue for WBC.

4.3.4 Keyed hash function

Some cryptographic algorithms use hash functions along with a secret key. For instance, message authentication codes (MAC) use a symmetric key, which is therefore expected to be protected. ISO/IEC 9797-2 specifies MAC algorithms which can benefit from WBC.

Another venue is that of key derivation functions (KDFs), when the derivation function is itself keyed. This derivation function can be white-boxed.

4.3.5 Customized cryptographic algorithm

The WBC technique can also advantageously be applied to a customized algorithm. If this algorithm is indeed kept secret (for example media protection with C2, SM1 or CSA, GSM A5, constants used in MILENAGE 4G telecommunication SIM (Subscriber Identification Module) management). The key extraction is a problem which is mixed with that of algorithm recovery. Indeed, it is usually considered hard to extract a key from an unknown algorithm (one noticeable exception is that of cube attacks).

Annex A gives justification for dedicated algorithms, which are easily amenable to be white-boxed. The specification of such algorithms can be (at least in parts) secret, so as to raise the difficulty of attacks.

4.4 Security properties

4.4.1 General

Security properties for the protection of the secret key of a WBC implementation are listed and described in 4.4. The two main security properties are described in 4.4.2 and 4.4.3, which correspond to the requirements on CSP in ISO/IEC 19790.

4.4.2 Secrecy of the key

The key is secret and the difficulty of secret extraction is increased over logically or physically accessible implementations that do not use such obfuscation measures. For instance, the attacker can analyse the device frontally, or instead deport its analysis on the white-boxing generator software which created it.

