Automated Verification of Block Cipher Modes of Operation, an Improved Method

Martin Gagné Pascal Lafourcade Yassine Lakhnech
Reihaneh Safavi-Naini

Abstract

In this paper, we improve on a previous result by Gagné et al. [11] for automatically proving the semantic security of symmetric modes of operation for block ciphers. While we use the same specification language programming language, we present a richer assertion language that uses more flexible invariants, and a more complete set of rules for establishing the invariants. In addition, all our invariants are given a meaningful semantic definition, whereas some invariants of the previous result relied on more ad hoc definitions. Our method can be used to verify the semantic security of all the encryption modes that could be proven secure in [11], in addition to a few new modes, such as Propagating Cipher-Block Chaining (PCBC).

1 Introduction

Block ciphers are among the most basic building blocks in cryptography. They can be used to construct primitives as varied as message authentication codes, hash functions and, their main application, symmetric encryption. Block ciphers are deterministic, and have fixed-size input and output, so protocols, called modes of operation, are required to encrypt messages of arbitrary length. The security of these modes of operation is then proven by reduction from the security of the mode of operation to some security property of the block cipher.

While the security of early modes of operation, such as Cipher Block Chaining (CBC), Cipher Feedback mode (CFB), Output Feedback (OFB), and Counter mode (CTR), can be proven relatively easily, the same cannot always be said for more recent modes of operation. Many new modes of operation were designed in the past decade (IACBC, IAPM [19], XCB [22], TMAC [18, 20], HCTR [6], HCH [8], EMU [15], EMU* [12], PEP [7], OMAC [16, 17], TET [13], CMC [14], GCM [23], EAX [5], XEX [24], TAE, TCH, TBC [21, 25] to name only a few), which often offer security properties that early modes did not possess. However, these additional properties often come at the cost of an increased complexity of the mode of operation, which, in turn, increase the complexity of the proof of security.

Automated verification tools can help increase our confidence in the security of these modes of operation by providing an independent argument for their security (or show us mistakes that had gone unnoticed). Gagné et al. [11] first initiated the study of automatic verification techniques for symmetric modes of operation. They presented an assertion language, invariants and rules for a Hoare logic which can be used to verify the security of most of the traditional modes of operation. However, due to the rather ad hoc nature of the description of certain invariants, and to the restrictiveness of their rule set, the resulting automated verifier was relatively limited.

Contributions: We improve on the result of Gagné et al. [11] by presenting a Hoare logic with a richer assertion language and invariants, which allow us to verify the security more modes of operation. For example, our new logic is able to verify the security of Propagating Cipher-Block Chaining (PCBC) – an encryption mode that was introduced for Kerberos version 4 – while [11] could not.

The programming language and assertion language are essentially the same as [11], but our invariants are much more precise. We use only three predicates: one that states that the value of a variable is indistinguishable from a random value, one that states that the block cipher has never been computed at the value of a variable, and one that keeps track of the most recent value of a counter. Our predicates are also much more satisfying than those in [11] since they can all be described using a clear semantic definition, whereas some predicates in [11] were rather ad hoc, particularly when it came to the predicate used to keep track of counters.

Using our logic as a set of rules for propagating the invariants though the code of each mode of operation, we can verify the semantic security of all the encryption modes which could be shown secure in [11], together with a few more, such as PCBC.

Related Work: **add ref to EasyCrypt** The security of symmetric encryption. An extensive discussion on different security notions for symmetric encryption and a proof of the CBC mode of encryption is presented in [3]. A security analysis of the encryption mode CBC-MAC is given in [4] and [26]. Many other works present other encryption modes with proofs of security.

Other works try to encode security of symmetric encryption modes as a non-interference property for programs with deterministic encryption. For example, [10] presents a computationally sound type system with exact security bounds for such programs. This type system has been applied to verify some symmetric encryption modes.

In [9], the authors proposed an automatic method for proving semantic security for asymmetric generic encryption schemes. A similar method is used in [11] to verify the security of symmetric encryption modes. Our work here is a continuation of these efforts.

Other works in automated verification of cryptographic protocols include [1], which presents a new logic for reasoning about cryptographic primitives, and uses this logic to formally analyze the security of the signature scheme PSS, and

[2], which provides a machine-checked proof of OAEP.

OUTLINE: Section 2 introduces some notation that will be used in the paper, as well as the definitions of concepts that will be used throughout the paper. In Section 3, we present our assertion language, invariants, and the set of rules that is used to propagate those invariants. We apply our method to some examples in Section 4 and conclude in Section 5.

2 Definitions

2.1 Notation and Conventions

Throughout this paper, we assume that all variables range over domains whose cardinality is exponential in the security parameter η. We also assume that all programs have length polynomial in η.

A block cipher is a family of permutations $\mathcal{E} : \{0,1\}^k \times \{0,1\}^\eta \to \{0,1\}^\eta$ indexed with a key $K \in \{0,1\}^k$. We often denote by $\mathcal{E}_K(x)$ the application of the block cipher with key K to the message block x. We generally omit the key used every time to simplify the notation, but it is understood that a key was selected at random at the beginning of the experiment and remains the same throughout.

For a mode of operation M, we denote by $\mathcal{E}_M$ the encryption function defined by M using block cipher $\mathcal{E}$.

For a probability distribution $\mathcal{D}$, we denote by $x \xleftarrow{\$} \mathcal{D}$ the operation of sampling a value x according to distribution $\mathcal{D}$. If S is a finite set, we denote by $x \xleftarrow{\$} S$ the operation of sampling x uniformly at random among the values in S.

Given two distribution ensembles $X = \{X_\eta\}_{\eta\in\mathbb{N}}$ and $X' = \{X'_\eta\}_{\eta\in\mathbb{N}}$, an algorithm $\mathcal{A}$ and $\eta \in \mathbb{N}$, we define the *advantage* of $\mathcal{A}$ in distinguishing X_η and X'_η as the following quantity:

$$\mathsf{Adv}(\mathcal{A}, \eta, X, X') = \mathsf{Pr}[x \xleftarrow{\$} X_\eta : \mathcal{A}(x) = 1] - \mathsf{Pr}[x \xleftarrow{\$} X'_\eta : \mathcal{A}(x) = 1].$$

Two distribution ensembles X and X' are called *indistinguishable*, denoted by $X \sim X'$, if $\mathsf{Adv}(\mathcal{A}, \eta, X, X')$ is negligible as a function of η for every probabilistic polynomial-time algorithm $\mathcal{A}$.

2.2 Grammar

We introduce our language for defining a generic encryption mode. All the encryption modes that will be verified by our method must be written using this language. The commands are given by the grammar of Figure 1, where:

- $x \xleftarrow{\$} \mathcal{U}$ denotes uniform sampling of a value and assigning it to x.

- $x := \mathcal{E}(y)$ denotes application of the block cipher $\mathcal{E}$ to the value of y and assigning the result to x.
- $x := y \oplus z$ denotes application of the exclusive-or operator to the values of y and z and assigning the result to x.
- $x := y||z$ represents the concatenation of the values of y and z.
- $x := y + 1$ increments by one the value of y and assigns the result to x. The operation is carried modulo 2^η
- $c_1; c_2$ is the sequential composition of c_1 and c_2

$$\begin{array}{lll} \mathrm{c} & ::= & x \xleftarrow{\$} \mathcal{U} \mid x := \mathcal{E}(y) \mid x := y \oplus z \mid x := y \| z \mid \\ & & \mid x := y + 1 \mid c_1; c_2 \end{array}$$

Figure 1: Language grammar

2.3 Generic Encryption Mode

We can now formally define a mode of encryption.

Definition 1 (Generic Encryption Mode)**.** *A generic encryption mode M is represented by $\mathcal{E}_M(m_1|\ldots|m_i, c_0|\ldots|c_i) : \mathbf{var}\ \vec{x_i}; \mathsf{c}_i$, where $\vec{x_i}$ is the set of variables used in c_i, all commands of c_i are built using the grammar described in Figure 1, each m_j is a message blocks, and each c_j is a cipher block, both of size n according to the input length of the block cipher $\mathcal{E}$.*

We add the additional block c_0 to the ciphertext because encryption modes are usually generate ciphertexts longer than the message. In all examples in this paper, c_0 will be the initialization vector (IV). The definition can easily be extended for encryption modes that also add one or more blocks at the end.

In Figure 2, we present the famous encryption mode $\mathcal{E}_{CBC}$ for a message of three blocks.

2.4 Semantics

In addition to the variables in Var,[1] we consider a variable $\mathcal{L}_\mathcal{E}$ that records the values on which $\mathcal{E}$ was computed. This variable remains hidden from the adversary. Thus, we consider states that assign bit-strings to the variables in Var and lists of pairs of bit-strings to $\mathcal{L}_\mathcal{E}$. Given a state S, $S(\mathcal{L}_\mathcal{E}).\mathsf{dom}$ and $S(\mathcal{L}_\mathcal{E}).\mathsf{res}$ denote the lists obtained by projecting each pair in $S(\mathcal{L}_\mathcal{E})$ to its first and second element respectively.

[1] We denote by Var the complete set of variables in the program, whereas **var** denotes the set of variables in the program that are not input or output variables.

$\mathcal{E}_{CBC}(m_1|m_2|m_3, IV|c_1|c_2|c_3)$:
var z_1, z_2, z_3;
$IV \xleftarrow{\$} \mathcal{U}$;
$z_1 := IV \oplus m_1$;
$c_1 := \mathcal{E}(z_1)$;
$z_2 := c_1 \oplus m_2$;
$c_2 := \mathcal{E}(z_2)$;
$z_3 := c_2 \oplus m_3$;
$c_3 := \mathcal{E}(z_3)$;

Figure 2: Description of $\mathcal{E}_{CBC}$

The state also contains a table $\mathcal{T}$, which is used to keep track of the variables used as counters. This table maintains a list of tables. Each new variable x whose value is sampled at random or obtained from a new query to the block cipher has its own list called $\mathcal{T}_x \in \mathcal{T}$ with $\mathcal{T}_x[0] = x$. For each i, $\mathcal{T}_x[i]$ will contain a variables whose value is known to be $x + i$, or the symbol $\perp$. For any variable y, we denote by $\mathcal{T}(y)$ the set $\{z \in \mathsf{Var} \mid \exists x, i, j \quad y = \mathcal{T}_x[i] \wedge z = \mathcal{T}_x[j]\}$, that is, the set of all variables that are in the same table as y. These tables are a history of variables with values are indistinguishable from random and their 'successors' obtained with the $+1$ operation. They will enable us to determine which values are used as counters, and which variables contain the most recent increment of a counter.

A program takes as input a *configuration* $(S, \mathcal{E}, \mathcal{T})$ and yields a distribution on configurations. A configuration is composed of a state S, a block cipher $\mathcal{E}$ and a table $\mathcal{T}$. Let $\Gamma_{\mathcal{E}}$ denote the set of configurations and $\textsc{Dist}(\Gamma_{\mathcal{E}})$ the set of distributions on configurations. The semantics is given in Table 1. In the table, $\delta(x)$ denotes the Dirac measure, i.e. $\mathsf{Pr}[x] = 1$ and $\mathcal{L}_{\mathcal{E}} \mapsto S(\mathcal{L}_{\mathcal{E}}) \cdot (x, y)$ denotes the addition of element (x, y) to $\mathcal{L}_{\mathcal{E}}$. Notice that the semantic function of commands can be lifted in the usual way to a function from $\textsc{Dist}(\Gamma_{\mathcal{E}})$ to $\textsc{Dist}(\Gamma_{\mathcal{E}})$. That is, let $\phi : \Gamma_{\mathcal{E}} \to \textsc{Dist}(\Gamma_{\mathcal{E}})$ be a function. Then, ϕ defines a unique function $\phi^* : \textsc{Dist}(\Gamma_{\mathcal{E}}) \to \textsc{Dist}(\Gamma_{\mathcal{E}})$ obtained by point-wise application of ϕ. By abuse of notation we also denote the lifted semantics by $[\![\mathsf{c}]\!]$.

A notational convention. It is easy to see that commands never alter $\mathcal{E}$. Therefore, we can, without ambiguity, write $S' \xleftarrow{\$} [\![c]\!](S, \mathcal{E})$ instead of $(S', \mathcal{E}) \xleftarrow{\$} [\![c]\!](S, \mathcal{E})$.

Here, we are only interested in distributions that can be constructed in polynomial time. We denote their set by $\textsc{Dist}(\Gamma, \mathcal{F})$, where $\mathcal{F}$ is a family of block ciphers, and is defined as the set of distributions of the form:

$$[\mathcal{E} \xleftarrow{\$} \mathcal{F}(1^\eta); S \xleftarrow{\$} [\![p]\!](I, \mathcal{E}) : (S, \mathcal{E})]$$

where p is a program with a polynomial number of commands, and I is the "initial" state, in which all variables are undefined and all lists and sets are

$$
\begin{array}{l}
[\![x \xleftarrow{\$} \mathcal{U}]\!](S,\mathcal{E}) = [u \xleftarrow{\$} \mathcal{U} : (S\{x \mapsto u, \mathcal{T} \mapsto \mathcal{T} \cup \{\mathcal{T}_x\}, \mathcal{E})] \\
[\![x := \mathcal{E}(y)]\!](S,\mathcal{E}) = \\
\quad \begin{cases} \delta(S\{x \mapsto v, \mathcal{T}, \mathcal{E}) \text{ if } (S(y), v) \in \mathcal{L}_{\mathcal{E}} \\ \delta(S\{x \mapsto v, \mathcal{T} \mapsto \mathcal{T} \cup \{\mathcal{T}_x\}, \mathcal{L}_{\mathcal{E}} \mapsto S(\mathcal{L}_{\mathcal{E}}) \cdot (S(y), v)\}, \mathcal{E}) \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{if } (S(y), v) \notin \mathcal{L}_{\mathcal{E}} \text{ and } v = \mathcal{E}(S(y)) \end{cases} \\
[\![x := y \oplus z]\!](S,\mathcal{E}) = \delta(S\{x \mapsto S(y) \oplus S(z), \mathcal{T}, \mathcal{E}) \\
[\![x := y||z]\!](S,\mathcal{E}) = \delta(S\{x \mapsto S(y)||S(z), \mathcal{T}, \mathcal{E}) \\
[\![x := y + 1]\!](S,\mathcal{E}) = \\
\quad \begin{cases} \delta(S\{x \mapsto S(y) + 1, \mathcal{T} \mapsto \mathcal{T} \cup \{\mathcal{T}_z \mapsto \mathcal{T}_z[i+1] = x\}, \mathcal{E}) \text{ if } y = \mathcal{T}_z[i] \wedge \mathcal{T}_z[i+1] = \perp \\ \delta(S\{x \mapsto S(y) + 1, \mathcal{T}, \mathcal{E}) \text{ otherwise} \end{cases} \\
[\![c_1; c_2]\!] = [\![c_2]\!] \circ [\![c_1]\!]
\end{array}
$$

Table 1: The semantics of the programming language

empty.

2.5 Security Model

Ideal Cipher Model

We prove the modes of encryption secure in the ideal cipher model. That is, we assume that the block cipher is a pseudo-random function.[2] This is a standard assumption for proving the security of any block-cipher-based scheme.

The advantage of an algorithm $\mathcal{A}$ against a family of pseudo-random function is defined as follows.

Definition 2. *Let $P : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a family of functions and let $\mathcal{A}$ be an algorithm that takes an oracle and returns a bit. The* prf-advantage *of $\mathcal{A}$ is defined as follows.*

$$\mathsf{Adv}^{prf}_{\mathcal{A},P} = Pr[K \xleftarrow{\$} \{0,1\}^k; \mathcal{A}^{P(K,\cdot)} = 1] - Pr[R \xleftarrow{\$} \Phi_n; \mathcal{A}^{R(\cdot)} = 1]$$

where Φ_n is the set of all functions from $\{0,1\}^n$ to $\{0,1\}^n$.

The security of a symmetric mode of operation is usually proven by first showing that the mode of operation would be secure if $\mathcal{E}$ was a random function in Φ_n. As a result, an adversary $\mathcal{A}$ against the encryption scheme can be transformed into an adversary $\mathcal{B}$ against the block cipher (as a pseudo-random function) with a similar running time, such that $\mathcal{B}$'s prf-advantage is similar to $\mathcal{A}$'s advantage in breaking the encryption scheme.

Encryption Security

Semantic security for a mode of encryption is defined as follows.

[2]While block ciphers are really families of permutations, it is well known that pseudo-random permutations are indistinguishable from pseudo-random functions if the block size is large enough.

Definition 3. *Let* $\mathcal{E}_M(m_1|\ldots|m_i, c_0|\ldots|c_i) : \mathbf{var}\ \vec{x_i}; \mathsf{c}_i$ *be a generic encryption mode.* $A = (A_1, A_2)$ *be an adversary and* $X \in \textsc{Dist}(\Gamma, \mathcal{E})$*. For* $\eta \in \mathbb{N}$*, let*

$$\begin{aligned}
\mathsf{Adv}^{ind-CPA}_{A,M} & (\eta, X) \\
= \quad & 2 * \mathsf{Pr}[(S, \mathcal{E}) \xleftarrow{\$} X; \\
& (x_0, x_1, p, s) \xleftarrow{\$} A_1^{\mathcal{O}_1}(\eta); b \xleftarrow{\$} \{0,1\}; \\
& S' \xleftarrow{\$} [\![\mathsf{c}_p]\!](S\{m_1|\ldots|m_p \mapsto x_b\}, \mathcal{E}) : \\
& A_2^{\mathcal{O}_2}(x_0, x_1, s, S'(c_0|\ldots|c_p)) = b] - 1
\end{aligned}$$

where $\mathcal{O}_1 = \mathcal{O}_2$ *are oracles that take a pair* (m, j) *as input, where* m *is a string and* j *is the block length of* m*, and answers using the* $j^t h$ *algorithm in* $\mathcal{E}_M$*.* A_1 *outputs* x_0, x_1 *such that* $|x_0| = |x_1|$ *and are composed of* p *blocks. The mode of operation* M *is semantically (IND-CPA) secure if* $\mathsf{Adv}^{ind-CPA}_{A,M}(\eta, X)$ *is negligible for any constructible distribution ensemble* X *and polynomial-time adversary* A*.*

It is important to note that in this definition, an adversary against the scheme is only given oracle access to the encryption mode $\mathcal{E}_M$, and *not* to the block cipher $\mathcal{E}$ itself.

Our method verifies the security of an encryption scheme by proving that the ciphertext is indistinguishable from random bits. It is a classical result that this implies semantic security.

3 Proving Semantic Security

In this section, we present our Hoare logic for proving semantic (IND-CPA) security for generic encryption mode defined with our language. We prove that our logic is sound although not complete. Our logic can be used to annotate each command of our programming language with a set of invariants that hold at each point of the program for any execution.

3.1 Assertion Language

We consider new predicates in order to consider properties of symmetric encryption modes. We use a Hoare Logic based on the following invariants:

$$\begin{aligned}
&\varphi ::= \mathsf{true} \mid \varphi \wedge \varphi \mid \psi \\
&\psi ::= \mathsf{Indis}(\nu x; V) \mid \mathsf{Indis}(\nu x; \mathcal{L}_{\mathcal{E}}) \mid \mathsf{E}(\mathcal{E}; x; V) \mid \mathsf{Icounter}(x; V) \mid
\end{aligned}$$

where $V \subseteq \mathsf{Var}$ and $x \in \mathsf{Var}$ is a variables used in the program. Intuitively:

Indis$(\nu x; V)$**:** means that no adversary has non-negligible probability to distinguish whether he is given results of computations performed using the value of x or a random value, when he is given the values of the variables in

V. In addition to variables in Var, the set V can contain a special variable $\mathcal{L}_\mathcal{E}$, in which case the invariant means that no adversary has non-negligible probability to distinguish whether he is given results of computations performed using the value of x or a random value, when he is given the values of the variables in V and $\mathcal{L}_\mathcal{E}.dom$.

$\mathsf{E}(\mathcal{E}; x; V)$: means the probability that the value of x is either in $\mathcal{L}_\mathcal{E}.dom$ or in V is negligible.

$\mathsf{Icounter}(x; V)$: means that x is the most recent value of a counter that started at a random value, and that the set V contains all the variables with previous values of the counter.

More formally, for each invariant ψ, we define that a distribution X satisfies ψ, denoted $X \models \psi$ as follows:

- $X \models \mathsf{true}$.
- $X \models \varphi \wedge \varphi'$ iff $X \models \varphi$ and $X \models \varphi'$.
- $X \models \mathsf{Indis}(\nu x; V)$ iff $[(S, \mathcal{E}) \xleftarrow{\$} X : (S(x, V), \mathcal{E})] \sim [(S, \mathcal{E}) \xleftarrow{\$} X; u \xleftarrow{\$} \mathcal{U}; S' = S\{x \mapsto u\} : (S'(x, V), \mathcal{E})]$ when $\mathcal{L}_\mathcal{E} \notin V$
- $X \models \mathsf{Indis}(\nu x; V)$ iff $[(S, \mathcal{E}) \xleftarrow{\$} X : (S(x, V \cup \mathcal{L}_\mathcal{E}.dom), \mathcal{E})] \sim [(S, \mathcal{E}) \xleftarrow{\$} X; u \xleftarrow{\$} \mathcal{U}; S' = S\{x \mapsto u\} : (S'(x, V \cup \mathcal{L}_\mathcal{E}.dom), \mathcal{E})]$ when $\mathcal{L}_\mathcal{E} \in V$
- $X \models \mathsf{Indis}(\nu x; \mathcal{L}_\mathcal{E})$ iff $[(S, \mathcal{E}) \xleftarrow{\$} X : (S(x) \cup S(\mathcal{L}_\mathcal{E}).dom, \mathcal{E})] \sim [(S, \mathcal{E}) \xleftarrow{\$} X; u \xleftarrow{\$} \mathcal{U}; S' = S\{x \mapsto u\} : (S'(x) \cup S'(\mathcal{L}_\mathcal{E}).dom, \mathcal{E})]$
- $X \models \mathsf{E}(\mathcal{E}; x; V)$ iff $\mathsf{Pr}[(S, \mathcal{E}) \xleftarrow{\$} X : S(x) \in S(\mathcal{L}_\mathcal{E}).\mathsf{dom} \cup S(V)]$ is negligible.
- $X \models \mathsf{Icounter}(x; V)$ iff $\mathsf{Indis}(x; \mathsf{Var} \setminus V)$ and $V = \mathcal{T}(x)$.

Notation: For a set V and a variable, we write V, x as a shorthand for $V \cup \{x\}$, $V - x$ as a shorthand for $V \setminus \{x\}$ and we write $\mathsf{Indis}(\nu x; V \cup \mathcal{L}_\mathcal{E})$ as a shorthand for $\mathsf{Indis}(\nu x; V) \wedge \mathsf{Indis}(\nu x; \mathcal{L}_\mathcal{E})$. We denote by Var^* the set $\mathsf{Var} \cup \mathcal{L}_\mathcal{E}$ and use $\mathsf{Indis}(\nu x)$ as a shorthand for $\mathsf{Indis}(\nu x; \mathsf{Var}^*)$.

Lemma 1. *The following are true for any set V and variables x, y with $x \neq y$*

1. $\mathit{Icounter}(x; V) \Rightarrow \mathit{Indis}(x; \mathit{Var} \setminus V)$
2. $\mathit{Indis}(\nu x; V \cup \mathcal{L}_\mathcal{E}) \Rightarrow \mathit{E}(\mathcal{E}; x; V \setminus \{x\})$
3. $\mathit{E}(\mathcal{E}; x; V) \wedge \mathit{Indis}(\nu x; \{y\}) \Rightarrow \mathit{E}(\mathcal{E}; x; V, y)$
4. $\mathit{Indis}(\nu x; V) \Rightarrow \mathit{Indis}(\nu x; V')$ *if* $V' \subset V$
5. $\mathit{E}(\mathcal{E}; x; V) \Rightarrow \mathit{E}(\mathcal{E}; x; V')$ *if* $V' \subset V$

Proof. 1. This clearly follows from the definition of $\mathsf{lcounter}(x; V)$.

2. We note that, for $u \xleftarrow{\$} \mathcal{U}$ and $S' = S\{x \mapsto u\}$, the set $S'(\mathcal{L}_\mathcal{E}).dom$ is equal to $S(\mathcal{L}_\mathcal{E}).com$ because $\mathcal{L}_\mathcal{E}$ is a list of values (not a list of variables). Therefore, if the probability that $S(x) \in S(\mathcal{L}_\mathcal{E}).dom$ is not negligible, it would be easy to differentiate between $S(x) \cup S(\mathcal{L}_\mathcal{E}).dom$ and $S'(x) \cup S'(\mathcal{L}_\mathcal{E}).dom$ because $S(x) \cup S(\mathcal{L}_\mathcal{E}).dom$ would contain one more collision than $S'(x) \cup S'(\mathcal{L}_\mathcal{E}).dom$ with non-negligible probability.

3. If $\mathsf{Indis}(\nu x; \{y\})$, then clearly $S(x) \neq S(y)$, otherwise it would be very easy to distinguish $S(\{x, y\})$ from $S'(\{x, y\})$ where $S' = S\{x \mapsto u\}$, $u \xleftarrow{\$} \mathcal{U}$. So from $\mathsf{E}(\mathcal{E}; x; V)$ and $\mathsf{Indis}(\nu x; \{y\})$, we have $S(x) \notin S(\mathcal{L}_\mathcal{E}).dom \cup S(V)$ and $S(x) \neq S(y)$. Thus, $S(x) \notin S(\mathcal{L}_\mathcal{E}).dom \cup S(V) \cup \{y\}$

4. If an algorithm can differentiate x from a random value given the set V', then clearly it could also differentiate x from a random value given the set V by simply ignoring the values that are in V but not in V'.

5. Clearly, $\mathsf{Pr}[(S, \mathcal{E}) \xleftarrow{\$} X : S(x) \in S(\mathcal{L}_\mathcal{E}).\mathsf{dom} \cup S(V')] \leq \mathsf{Pr}[(S, \mathcal{E}) \xleftarrow{\$} X : S(x) \in S(\mathcal{L}_\mathcal{E}).\mathsf{dom} \cup S(V)]$ since $V' \subset V$

□

3.2 Hoare Logic Rules

We present a set of rules of the form $\{\varphi\}\mathsf{c}\{\varphi'\}$, meaning that execution of command c in any distribution that satisfies φ leads to a distribution that satisfies φ'. Using Hoare logic terminology, this means that the triple $\{\varphi\}\mathsf{c}\{\varphi'\}$ is valid. We group rules together according to their corresponding commands. those commands are only used during decryption.

In all the rules below, unless indicated otherwise, we assume that $t \notin \{x, y, z\}$ and $x \notin \{y, z\}$. In addition, for all rules involving the invariant Indis, $\mathcal{L}_\mathcal{E}$ can be one of the elements in the set V.

Random Assignment:

- (R1) $\{true\}\ x \xleftarrow{\$} \mathcal{U}\ \{\mathsf{Indis}(\nu x) \wedge \mathsf{lcounter}(x; \{x\})\}$
- (R2) $\{\mathsf{Indis}(\nu t; V)\}\ x \xleftarrow{\$} \mathcal{U}\ \{\mathsf{Indis}(\nu t; V, x)\}$
- (R3) $\{\mathsf{E}(\mathcal{E}; t; V)\}\ x \xleftarrow{\$} \mathcal{U}\ \{\mathsf{E}(\mathcal{E}; t; V, x)\}$

Increment:

- (I1) $\{\mathsf{lcounter}(y; V)\}\ x := y + 1\ \{\mathsf{lcounter}(x; V, x) \wedge \mathsf{E}(\mathcal{E}; x; \mathsf{Var} - x)\}$
- (I2) $\{\mathsf{Indis}(\nu y; V)\}\ x := y + 1\ \{\mathsf{Indis}(\nu x; V)\}$ if $y \notin V$
- (I3) $\{\mathsf{Indis}(\nu t; V)\}\ x := y + 1\ \{\mathsf{Indis}(\nu t; V)\}$ if $x \notin V$ even if $t = y$
- (I4) $\{\mathsf{Indis}(\nu t; V, y)\}\ x := y + 1\ \{\mathsf{Indis}(\nu t; V, x, y)\}$ if $x \notin V$

- (I5) $\{\mathsf{Icounter}(y; V_1) \wedge \mathsf{E}(\mathcal{E}; t; V_2)\}\ x := y + 1\ \{\mathsf{E}(\mathcal{E}; t; V_2, x)\}$ even if $t = y$

Xor operator:

- (X1) $\{\mathsf{Indis}(\nu y; V, y, z)\}\ x := y \oplus z\ \{\mathsf{Indis}(\nu x; V, x, z)\}$ if $y \neq z$ and $y \notin V$
- (X2) $\{\mathsf{Indis}(\nu t; V)\}\ x := y \oplus z\ \{\mathsf{Indis}(\nu t; V)\}$ if $x \notin V$, even if $t = y$ or $t = z$
- (X3) $\{\mathsf{Indis}(\nu t; V, y, z)\}\ x := y \oplus z\ \{\mathsf{Indis}(\nu t; V, x, y, z)\}$

Due to the commutativity of the Xor operation, the role of y and z can be reversed in all the rules above.

Block cipher:

- (B1) $\{\mathsf{E}(\mathcal{E}; y; \emptyset)\}\ x := \mathcal{E}(y)\ \{\mathsf{Indis}(\nu x) \wedge \mathsf{Icounter}(x; \{x\})\}$
- (B2) $\{\mathsf{Indis}(\nu t; V) \wedge \mathsf{E}(\mathcal{E}; y; \emptyset)\}\ x := \mathcal{E}(y)\ \{\mathsf{Indis}(\nu t; V, x)\}$ provided $\mathcal{L}_{\mathcal{E}} \notin V$
- (B3) $\{\mathsf{Indis}(\nu t; \{\mathcal{L}_{\mathcal{E}}, y\})\}\ x := \mathcal{E}(y)\ \{\mathsf{Indis}(\nu t; \mathcal{L}_{\mathcal{E}})\}$
- (B4) $\{\mathsf{Icounter}(t; V)\}\ x := \mathcal{E}(y)\ \{\mathsf{Icounter}(t; V)\}$ even if $t = y$
- (B5) $\{\mathsf{E}(\mathcal{E}; t; V, y)\}\ x := \mathcal{E}(y)\ \{\mathsf{E}(\mathcal{E}; t; V, y)\}$

Concatenation:

- (C1) $\{\mathsf{Indis}(\nu y; V, y, z) \wedge \mathsf{Indis}(\nu z; V, y, z)\}\ x := y \| z\ \{\mathsf{Indis}(\nu x; V, x)\}$ if $y, z \notin V$
- (C2) $\{\mathsf{Indis}(\nu t; V, y, z)\}\ x := y \| z\ \{\mathsf{Indis}(\nu t; V, x, y, z)\}$
- (C3) $\{\mathsf{Indis}(\nu t; V)\}\ x := y \| z\ \{\mathsf{Indis}(\nu t; V)\}$ if $x \notin V$, even if $t = y$ or $t = z$

Finally, we add a few rules whose purpose is to preserve invariants that are unaffected by the command.

Generic preservation rules:
Assume that $t \neq x, y, z$ and c is either $x \xleftarrow{\$} \mathcal{U}$, $x := y \| z$, $x := y \oplus z$, or $x := w + 1$:

- (G1) $\{\mathsf{Icounter}(t; V)\}\ \mathsf{c}\ \{\mathsf{Icounter}(\mathrm{t};\mathrm{V})\}$ if $y, z \notin V$
- (G2) $\{\mathsf{E}(\mathcal{E}; t; V)\}\ \mathsf{c}\ \{\mathsf{E}(\mathcal{E}; t; V)\}$ if $x \notin V$, even if $t = y$ or $t = z$

3.3 Minimality of the Rule Set

We now show that our rule set is minimal: whenever a rule has more than one precondition, both preconditions are needed, and if a rule creates two invariants, neither implies the other.

- (R1) $\{true\}\ x \xleftarrow{\$} \mathcal{U}\ \{\mathsf{Indis}(\nu x) \wedge \mathsf{E}(\mathcal{E}; x)\}$

1. $\mathsf{Indis}\nu x$ does not imply $\mathsf{E}(\mathcal{E}; x)$, which can be seen in the following program:

$$x \xleftarrow{\$} \mathcal{U}; \quad y := \mathcal{E}(x)$$

   At the end of this program, $\mathsf{Indis}(\nu x)$ still holds, but clearly, $\mathsf{E}(\mathcal{E}; x)$ does not since x has been queried to $\mathcal{E}$.

2. $\mathsf{E}(\mathcal{E}; x)$ does not imply $\mathsf{Indis}(\nu x)$, which can be seen in the following program:

$$x \xleftarrow{\$} \mathcal{U}; \quad y := x + 1$$

   At the end of the program, $\mathsf{E}(\mathcal{E}; x)$ still holds, but $\mathsf{Indis}\nu x$ does not, since x can easily be distinguished from a random value when given the value of y.

- (X1) $\{\mathsf{Indis}(\nu y; V, y, z)\}\ x := y \oplus z\ \{\mathsf{Indis}(\nu x; V, x, z)\}$

The same counterexample as for the previous rule can be used to show that this rule is minimal.

- (X4) $\{\mathsf{Indis}(x) \wedge \mathsf{E}(\mathcal{E}; x)\}\ z := x \oplus y\ \{\mathsf{E}(\mathcal{E}; z)\}$ if $y \neq z$

1. the rule "$\{\mathsf{Indis}(x)\}\ z := x \oplus y\ \{\mathsf{E}(\mathcal{E}; z)\}$ if $y \neq z$" is false, as can be seen in the following program:

$$x \xleftarrow{\$} \mathcal{U}; t \xleftarrow{\$} \mathcal{U}; \quad y := t \oplus t; \quad w := \mathcal{E}(x); \quad z := x \oplus y.$$

   We can easily see that before the last command, $\mathsf{Indis}(\nu x)$ still holds, but $\mathsf{E}(\mathcal{E}; z)$ does not hold after the program because the value in x and the value in z are the same, and x has been queried to $\mathcal{E}$.

2. the rule "$\{\mathsf{E}(\mathcal{E}; x)\}\ z := x \oplus y\ \{\mathsf{E}(\mathcal{E}; z)\}$ if $y \neq z$" is false, as can be seen in the following program:

$$x \xleftarrow{\$} \mathcal{U}; \quad y \xleftarrow{\$} \mathcal{U}; \quad t := x \oplus y; \quad u := \mathcal{E}t; \quad z := x \oplus y$$

   Clearly, $\mathsf{E}(\mathcal{E}; x)$ holds throughout the program since x is sampled at random, but $\mathsf{E}(\mathcal{E}; z)$ does not hold at the end of the program because the values in t and z are equal and t has been queried to the block cipher.

- (B2) $\{\mathsf{E}(\mathcal{E}; x) \wedge \mathsf{E}(\mathcal{E}; z)\}\ y := \mathcal{E}(x)\ \{\mathsf{E}(\mathcal{E}; z)\}$ provided $z \neq x$

1. The rule "$\{\mathsf{E}(\mathcal{E}; z)\}\ y := \mathcal{E}(x)\ \{\mathsf{E}(\mathcal{E}; z)\}$ provided $z \neq x$" is false, as can be seen in the following program:

$$x \xleftarrow{\$} \mathcal{U}; \quad w \xleftarrow{\$} \mathcal{U}; \quad y := w \oplus w; \quad z := x \oplus y; \quad t := \mathcal{E}(x)$$

2. The rule "$\{\mathsf{E}(\mathcal{E}; x)\}\ y := \mathcal{E}(x)\ \{\mathsf{E}(\mathcal{E}; z)\}$ provided $z \neq x$" is clearly false. The command $x := \mathcal{E}(y)$ does not involve z, and cannot introduce the property $\{\mathsf{E}(\mathcal{E}; z)\}$ about every variable.

- (B3) $\{\mathsf{E}(\mathcal{E}; x) \wedge \mathsf{Indis}(z, v)\}\ y := \mathcal{E}(x)\ \{\mathsf{Indis}(z; V, y)\}$ provided $z \neq x$

1. The rule "$\{\mathsf{Indis}(z, v)\}\ y := \mathcal{E}(x)\ \{\mathsf{Indis}(z; V, y)\}$ provided $z \neq x$" is false, as can be seen in the following program:

$$x \xleftarrow{\$} \mathcal{U}; \quad w \xleftarrow{\$} \mathcal{U}; \quad y := w \oplus w; \quad z := x \oplus y; \quad t := \mathcal{E}(z); \quad u := \mathcal{E}x$$

2. the rule "$\{\mathsf{E}(\mathcal{E}; x)\}\ y := \mathcal{E}(x)\ \{\mathsf{Indis}(\nu z; V, y)\}$ provided $z \neq x$" is clearly false. The command $x := \mathcal{E}(y)$ does not involve z, and cannot introduce the property $\{\mathsf{Indis}(\nu z; V, y)\}$ about every variable.

- (B4) $\{\mathsf{E}(\mathcal{E}; y) \wedge \mathsf{lcounter}(z)\}\ x := \mathcal{E}(y)\ \{\mathsf{lcounter}(z)\}$ (even when $z = y$)

The same counterexamples as for the previous rule can be used to show that this rule is minimal since $\mathsf{Indis}(\nu x)$ implies $\mathsf{lcounter}(x)$.

4 Examples

As in [11], we apply our method to the traditional encryption modes (CBC), (CFB), (OFB) and (CTR) in Figure 3, 4, 5 and 6 respectively. We also show how we can verify the security of the mode PCBC, which could not be proven secure in [11]. For simplicity, we consider messages consisting of only 3 blocks, as it should be clear that the same propagation of invariants would continue for any finite number of blocks. In order to prove IND-CPA security of these encryption schemes we have to prove that $c_0 = IV, c_1, c_2, c_3$ are indistinguishable from random bit strings when given $m_1, m_2, m_3, c_0, c_1, c_2$ and c_3. For the sake of readability, we will only show the invariants that are necessary for the proof of security. An automated verifier would however derive every possible invariant by applying every applicable rule at each step. In all the examples, we denote by, say, (L3) the use of the third part of Lemma 1.

CBC & CFB : In Figure 3 and 4, we describe the application of our set of rules on CBC and CFB examples. The analysis of these two encryption modes are similar.

OFB : Note that, contrary to [11], we no longer need to put the commands describing this mode of operation in a peculiar order to be able to verify its security, we can now verify OFB with the commands put in the more 'natural' order.

CTR : This scheme is the only one of the four encryption modes we have studied that uses the increment command. The analysis is presented in Figure 6. We can see how the $\mathsf{lcounter}$ invariant is used for proving the IND-CPA security of this mode.

$\mathcal{E}_{CBC}(m_1|m_2|m_3, IV|c_1|c_2|c_3)$
var IV, z_1, z_2, z_3;

$IV \xleftarrow{\$} \mathcal{U}$;	$\{\mathsf{Indis}(\nu IV)\}$	(R1)
$z_1 := IV \oplus m_1$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var}^* - z_1) \wedge \mathsf{Indis}(\nu z_1; \mathsf{Var}^* - IV)$	(X1)(X2)
	$\wedge\ \mathsf{E}(\mathcal{E}; z_1; \mathsf{Var} - IV - z_1)\}$	(L2)
$c_1 := \mathcal{E}(z_1)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - z_1) \wedge \mathsf{Indis}(\nu c_1)\}$	(B1)(B2)
$z_2 := c_1 \oplus m_2$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - z_1) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var}^* - z_2)$	(X2)(X3)
	$\wedge\ \mathsf{Indis}(\nu z_2; \mathsf{Var}^* - c_1) \wedge \mathsf{E}(\mathcal{E}; z_2; \mathsf{Var} - c_1 - z_2)\}$	(X1)(L2)
$c_2 := \mathcal{E}(z_2)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - z_1) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_2)$	(B2)
	$\wedge\ \mathsf{Indis}(\nu c_2)\}$	(B1)
$z_3 := c_2 \oplus m_3$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - z_1) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_2)$	(X3)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var}^* - z_3) \wedge \mathsf{Indis}(\nu z_3; \mathsf{Var}^* - c_2)$	(X1)(X2)
	$\wedge\ \mathsf{E}(\mathcal{E}; z_3; \mathsf{Var} - c_2 - z_3)\}$	(L2)
$c_3 := \mathcal{E}(z_3)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - z_1) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_2)$	(B2)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var} - z_3) \wedge Indis(\nu c_3; \mathsf{Var})\}$	(B1)

Figure 3: Analysis of CBC encryption mode

$\mathcal{E}_{CFB}(m_1|m_2|m_3, IV|c_1|c_2|c_3)$
var IV, z_1, z_2, z_3;

$IV \xleftarrow{\$} \mathcal{U}$;	$\{\mathsf{Indis}(\nu IV) \wedge \mathsf{E}(\mathcal{E}; IV; \mathsf{Var} - IV)\}$	(R1)(L1)
$z_1 := \mathcal{E}(IV)$;	$\{\mathsf{Indis}(\nu IV) \wedge \mathsf{Indis}(\nu z_1)\}$	(B1)(B2)
$c_1 := z_1 \oplus m_1$;	$\{\mathsf{Indis}(\nu IV) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var}^* - z_1)$	(X1)(X3)
	$\wedge \mathsf{E}(\mathcal{E}; c_1; \mathsf{Var} - z_1 - c_1)\}$	(L1)
$z_2 := \mathcal{E}(c_1)$;	$\{\mathsf{Indis}(\nu IV) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1)\}$	(B2)
	$\wedge\ \mathsf{Indis}(\nu z_2)\}$	(B1)
$c_2 := z_2 \oplus m_2$;	$\{\mathsf{Indis}(\nu IV) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1)$	(X3)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var}^* - z_2) \wedge \mathsf{E}(\mathcal{E}; c_2; \mathsf{Var} - z_2 - c_2)\}$	(X1)(L4)
$z_3 := \mathcal{E}(c_2)$;	$\{\mathsf{Indis}(\nu IV) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1)$	(B2)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var} - z_2) \wedge \mathsf{Indis}(\nu z_3)\}$	(B1)
$c_3 := z_3 \oplus m_3$;	$\{\mathsf{Indis}(\nu IV) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1)$	(X3)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var} - z_2) \wedge \mathsf{Indis}(\nu c_3; \mathsf{Var}^* - z_3)\}$	(X1)

Figure 4: Analysis of CFB encryption mode

PCBC : The security of this scheme could not be verified by [11] because it is necessary to be able to infer that the value of certain variable has not been queried to the block cipher even after two Xor operations. Our new set of rules can verify its security as can be seen in Figure 7.

5 Conclusion

We improved on the result of Gagné et al. [11] by proposing a new Hoare logic with more precise invariants and more complete rule set. This logic can be used to construct an automated verification tool that can successfully verify the security of all the symmetric encryption modes that could be verified by [11], in addition to many more that it could not.

Future directions to this work include the addition of loops to our grammar to remove the necessity of having a different program for each message length. We would also like to use a similar system to model other security properties, such as unforgeability and collision-resistance. We believe that the unforgeability property (IND-CTXT) would be of particular interest since, combined with semantic security, it would allow us to prove the chosen-ciphertext (CCA) security of certain symmetric encryption modes.

References

[1] Gilles Barthe, Marion Daubignard, Bruce Kapron, and Yassine Lakhnech. Computational indistinguishability logic. In *Proceedings of the 17th ACM conference on Computer and communications security*, CCS '10, pages 375–386. ACM, 2010.

[2] Gilles Barthe, Benjamin Grégoire, Yassine Lakhnech, and Santiago Zanella Béguelin. Beyond provable security verifiable ind-cca security of oaep. In *CT-RSA*, Lecture Notes in Computer Science, pages 180–196. Springer, 2011.

[3] Mihir Bellare, Anand Desai, Eron Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:394, 1997.

[4] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of the cipher block chaining message authentication code. *J. Comput. Syst. Sci.*, 61(3):362–399, 2000.

[5] Mihir Bellare, Phillip Rogaway, and David Wagner. The EAX mode of operation. In Bimal K. Roy and Willi Meier, editors, *FSE*, volume 3017 of *Lecture Notes in Computer Science*, pages 389–407. Springer, 2004.

[6] Debrup Chakraborty and Mridul Nandi. An improved security bound for HCTR. In *Fast Software Encryption: 15th International Workshop, FSE*

2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers, pages 289–302, Berlin, Heidelberg, 2008. Springer-Verlag.

[7] Debrup Chakraborty and Palash Sarkar. A new mode of encryption providing a tweakable strong pseudo-random permutation. In Matthew J. B. Robshaw, editor, *FSE*, volume 4047 of *Lecture Notes in Computer Science*, pages 293–309. Springer, 2006.

[8] Debrup Chakraborty and Palash Sarkar. HCH: A new tweakable enciphering scheme using the hash-counter-hash approach. *IEEE Transactions on Information Theory*, 54(4):1683–1699, 2008.

[9] Judicael Courant, Marion Daubignard, Cristian Ene, Pascal Lafourcade, and Yassine Lahknech. Towards automated proofs for asymmetric encryption schemes in the random oracle model. In *Proceedings of the 15th ACM Conference on Computer and Communications Security, (CCS'08)*, Alexandria, USA, October 2008.

[10] Judicaël Courant, Cristian Ene, and Yassine Lakhnech. Computationally sound typing for non-interference: The case of deterministic encryption. In Vikraman Arvind and Sanjiva Prasad, editors, *FSTTCS 2007: Foundations of Software Technology and Theoretical Computer Science, 27th International Conference, New Delhi, India, December 12-14, 2007, Proceedings*, volume 4855 of *Lecture Notes in Computer Science*, pages 364–375. Springer, 2007.

[11] Martin Gagné, Pascal Lafourcade, Yassine Lakhnech, and Reihaneh Safavi-Naini. Automated proofs for encryption modes. In *13th Annual Asian Computing Science Conference Focusing on Information Security and Privacy: Theory and Practice (ASIAN'09)*, volume 5913 of *LNCS*, pages 39–53, 2009.

[12] Shai Halevi. EME*: Extending EME to handle arbitrary-length messages with associated data. In Anne Canteaut and Kapalee Viswanathan, editors, *Progress in Cryptology - INDOCRYPT 2004, 5th International Conference on Cryptology in India, Chennai, India, December 20-22, 2004, Proceedings*, volume 3348 of *Lecture Notes in Computer Science*, pages 315–327. Springer, 2004.

[13] Shai Halevi. Invertible universal hashing and the tet encryption mode. In Alfred Menezes, editor, *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 412–429. Springer, 2007.

[14] Shai Halevi and Phillip Rogaway. A tweakable enciphering mode. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 482–499. Springer, 2003.

[15] Shai Halevi and Phillip Rogaway. A parallelizable enciphering mode. In Tatsuaki Okamoto, editor, *CT-RSA*, volume 2964 of *Lecture Notes in Computer Science*, pages 292–304. Springer, 2004.

[16] Tetsu Iwata and Kaoru Kurosawa. OMAC: One-key CBC MAC. In Thomas Johansson, editor, *FSE*, volume 2887 of *Lecture Notes in Computer Science*, pages 129–153. Springer, 2003.

[17] Tetsu Iwata and Kaoru Kurosawa. On the security of a new variant of OMAC. In Jong In Lim and Dong Hoon Lee, editors, *ICISC*, volume 2971 of *Lecture Notes in Computer Science*, pages 67–78. Springer, 2003.

[18] Tetsu Iwata and Kaoru Kurosawa. Stronger security bounds for OMAC, TMAC, and XCBC. In Thomas Johansson and Subhamoy Maitra, editors, *INDOCRYPT*, volume 2904 of *Lecture Notes in Computer Science*, pages 402–415. Springer, 2003.

[19] Charanjit S. Jutla. Encryption modes with almost free message integrity. In *EUROCRYPT '01: Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, pages 529–544, London, UK, 2001. Springer-Verlag.

[20] Kaoru Kurosawa and Tetsu Iwata. TMAC: Two-key CBC MAC. In Marc Joye, editor, *CT-RSA*, volume 2612 of *Lecture Notes in Computer Science*, pages 33–49. Springer, 2003.

[21] Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable block ciphers. In *CRYPTO '02: Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, pages 31–46, London, UK, 2002. Springer-Verlag.

[22] David A. McGrew and Scott R. Fluhrer. The security of the extended codebook (xcb) mode of operation. In Carlisle M. Adams, Ali Miri, and Michael J. Wiener, editors, *Selected Areas in Cryptography*, volume 4876 of *Lecture Notes in Computer Science*, pages 311–327. Springer, 2007.

[23] David A. McGrew and John Viega. The security and performance of the galois/counter mode (GCM) of operation. In Anne Canteaut and Kapalee Viswanathan, editors, *INDOCRYPT*, volume 3348 of *Lecture Notes in Computer Science*, pages 343–355. Springer, 2004.

[24] Phillip Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In Pil Joong Lee, editor, *ASIACRYPT*, volume 3329 of *Lecture Notes in Computer Science*, pages 16–31. Springer, 2004.

[25] Peng Wang, Dengguo Feng, and Wenling Wu. On the security of tweakable modes of operation: TBC and TAE. In Jianying Zhou, Javier Lopez, Robert H. Deng, and Feng Bao, editors, *ISC*, volume 3650 of *Lecture Notes in Computer Science*, pages 274–287. Springer, 2005.

[26] Éliane Jaulmes, Antoine Joux, and Frédéric Valette. On the security of randomized CBC-MAC beyond the birthday paradox limit - a new construction. In *Fast Software Encryption 02, Lecture Notes in Computer Science*, pages 237–251. Springer-Verlag, 2001.

A Proofs

Notations: We write $\nu x \cdot X$ to denote the distribution

$$[v \xleftarrow{\$} \mathcal{U}; (S, \vec{E}) \xleftarrow{\$} X : (S\{x \mapsto v\}, \vec{E})],$$

where $\mathcal{U}$ is the uniform distribution on the values on which x ranges. Let X be a family of distributions in $\textsc{Dist}(\Gamma, \vec{E})$ and V be a set of variables in Var. By $D(X, V)$ we denote the following distribution family (on tuples of bit-strings): $D(X, V)_\eta = [(S, \mathcal{E}) \xleftarrow{\$} X : (S(V), \vec{E})]$. Here $S(V)$ is the point-wise application of S to the elements of V. We say that X and X' are *V-indistinguishable*, denoted by $X \sim_V X'$, if $D(X, V) \sim D(X', V)$. Hence we have $X \models \mathsf{Indis}(\nu x; V_1)$ iff $X \sim_{V_1} \nu x \cdot X$ which is equivalent to the definition given in Section 3.

Lemma 2. *For any $X, X' \in \textsc{Dist}(\Gamma, \vec{E})$, any set of variables V, any expression e constructible from V, and any variable x, if $X \sim_V X'$ then $[\![x := e]\!](X) \sim_{V,x} [\![x := e]\!](X')$.*

Proof. We assume $X \sim_V X'$. If we suppose that $[\![x := e]\!](X) \not\sim_{V,x} [\![x := e]\!](X')$, then there exists A a poly-time adversary that, on input V, x drawn either from $[\![x := e]\!](X)$ or $[\![x := e]\!](X')$, guesses the right initial distribution with non-negligible probability. We let B be the following adversary against $X \sim_V X'$: $B(V)$:= let $x := e$ in $A(V, x)$. The idea is that B can evaluate in polynomial time the expression e using its own inputs. Hence it can provide the appropriate inputs to A. It is clear that the advantage of B is exactly that of A, which would imply that it is not negligible, although we assumed $X \sim_V X'$. □

Corollary 3. *For any $X \in \textsc{Dist}(\Gamma, \vec{E})$, any sets of variables V, any expression e constructible from V, and any variable x, z such that $z \notin \{x\} \cup \mathsf{Var}(e)$ if $X \models \mathsf{Indis}(\nu z; V)$ then $[\![x := e]\!](X) \models \mathsf{Indis}(\nu z; V)$. We emphasize that here we use the notation $\mathsf{Var}(e)$ (in its usual sense), that is to say, the variable z does not appear* at all *in e.*

Proof. $X \models \mathsf{Indis}(\nu z; V)$ is equivalent to $X \sim_V \nu z.X$. Using Lemma 2 we get $[\![x := e]\!](X) \sim_V [\![x := e]\!](\nu z.X)$. Since $z \notin \{x\} \cup \mathsf{Var}(e)$ we have that $[\![x := e]\!](\nu z.X) = \nu z.[\![x := e]\!](X)$ and hence $[\![x := e]\!](X) \sim_{V,x} \nu z.[\![x := e]\!](X)$, that is $[\![x := e]\!](X) \models \mathsf{Indis}(\nu z; V, x)$. □

Lemma 4. *For any $X, X' \in \textsc{Dist}(\Gamma, \vec{E})$, any sets of variables V, and any variable x, if $X \sim_V X'$ then $X \models \mathsf{Indis}(\nu x; V) \iff X' \models \mathsf{Indis}(\nu x; V)$.*

Proof. By symmetry of indistinguishability and equivalence, the conclusion follows from a single implication. We assume $X \sim_V X'$. Hence, $\nu x.X \sim_V \nu x.X'$; this can be justified by an immediate reduction. Moreover, the hypothesis $X \models \mathsf{Indis}(\nu x; V)$ implies $X \sim_V \nu x.X$. By transitivity of the indistinguishability relation, we get $X' \sim_V \nu x.X'$. Thus, $X' \models \mathsf{Indis}(\nu x; V)$. □

A.0.1 Random Assignment

Proposition 5 (Rule R1). $\{true\}\ x \xleftarrow{\$} \mathcal{U}\ \{\mathit{Indis}(\nu x) \wedge \mathit{Icounter}(x;\{x\})\}$

Proof. The proof of this rule can be split in two parts $\{true\}\ x \xleftarrow{\$} \mathcal{U}\ \{\mathsf{Indis}(\nu x)\}$ and $\{true\}\ x \xleftarrow{\$} \mathcal{U}\ \{\mathsf{Icounter}(x;\{x\})\}$. The first rule is immediate from the definitions of the indistinguishability predicate. As for the second rule, clearly, $\mathsf{Indis}(\nu x; \mathsf{Var} - x)$ is immediate, and looking at the semantics, we find that after the command $x \xleftarrow{\$} \mathcal{U}$, $\mathcal{T}(x) = \{x\}$, which shows that $\mathsf{Icounter}(x;\{x\})$ holds. □

Proposition 6 (Rule R2). $\{\mathit{Indis}(\nu t; V)\}\ x \xleftarrow{\$} \mathcal{U}\ \{\mathit{Indis}(\nu t; V, x)\}$

Proof. The intuition is that x being completely random, providing its value to the adversary does not help this latter in any way. We show the result by reduction. Assume that there exists an adversary B against $[\![x \xleftarrow{\$} \mathcal{U}]\!](X) \models \mathsf{Indis}(\nu t; V, x)$ that can distinguish with non-negligible advantage between t and a random value given the values of V and x. Then, we can construct an adversary $A(V)$ playing against $X \models \mathsf{Indis}(\nu y; V)$ that has the same advantage as B: $A(V)$ draws a value u at random and runs $B(V)$, and then returns B's answer. If B has non-negligible advantage, then so does A, which contradicts our hypothesis. □

Proposition 7 (Rule R3). $\{\mathit{E}(\mathcal{E}; t; V)\}\ x \xleftarrow{\$} \mathcal{U}\ \{\mathit{E}(\mathcal{E}; t; V, x)\}$

Proof. Before the command is executed, we have that the probability that $S(t) \in S(\mathcal{L}_\mathcal{E}).dom \cup S(V)$ is negligible. Also, by hypothesis, the domain from which x is being sampled is of exponential size in the security parameter, so the probability that $S(t) = S(x)$ is negligible. Thus, the probability that $S(t) \in S(\mathcal{L}_\mathcal{E}).dom \cup S(V) \cup \{x\}$ is negligible as well. □

A.1 Increment

Proposition 8 (Rule I1). $\{\mathit{Icounter}(y; V)\}\ x := y + 1\ \{\mathit{Icounter}(x; V, x) \wedge \mathit{E}(\mathcal{E}; x; \mathit{Var} - x)\}$

Proof. The proof of this rule is split in 2 parts:
$\{\mathsf{Icounter}(y; V)\}\ x := y + 1\ \{\mathsf{Icounter}(x; V, x)\}$
$\{\mathsf{Icounter}(y; V)\}\ x := y + 1\ \{\mathsf{E}(\mathcal{E}; x; \mathsf{Var} - x)\}$.
For the first part, we know that before the command is executed, we have $\{\mathsf{Indis}(\nu y; \mathsf{Var} \setminus V)\}$, and $V = \mathcal{T}(y)$ by definition of $\mathsf{Icounter}(y; V)$. From the semantics, we easily find that after the command is executed, $\mathcal{T}(x) = \mathcal{T}(y) \cup \{x\}$. Suppose that $\{\mathsf{Indis}(\nu x; \mathsf{Var} \setminus (V \cup \{x\})\}$ does not hold after the command. Then there would exist an algorithm $\mathcal{B}$ which, given the values in $S(\mathsf{Var} \setminus (V \cup \{x\}))$ can differentiate between the value of x and a random value. Using this algorithm, we could construct an algorithm $\mathcal{A}$ that distinguishes y from a random value given the values in $\mathsf{Var} \setminus V$ as follows: $\mathcal{A}(S(y), S(\mathsf{Var} \setminus V))$ simply runs $\mathcal{B}(S(y) + 1, S(\mathsf{Var} \setminus (V \cup \{x\})))$ and returns its answer. This contradicts our assumption

that $\{\mathsf{Indis}(\nu y; \mathsf{Var} \setminus V)\}$. Therefore, we have that $\{\mathsf{Indis}(\nu x; \mathsf{Var} \setminus (V \cup \{x\})\}$ holds after the command is executed, hence $\{\mathsf{Icounter}(x; V, x)\}$ holds.

For the second part, first note that since the value of x is independent from the value of all variables other than $\mathcal{T}(x)$, we have that the probability that the value of x is equal to any value in $\mathsf{Var} - \mathcal{T}(x)$ is negligible. Also, since $S(x) = S(t) + k$ for all $t \in \mathcal{T}(x)$ for $k \neq 0$, we also have that the value of x is different from the value of any variable in $\mathcal{T}(x)$ save for x itself. Therefore, the probability that $S(x) \in S(\mathsf{Var} - x)$ is negligible. Next, we know that since the starting point in $\mathcal{T}(x)$ was random and independent from all other values in the system, the only value in $\mathcal{L}_\mathcal{E}$ from which the value of x may not be independent are those values in $\mathcal{T}(x)$ on which the block cipher may have been computed. Thus, the probability that the value of x is already in $\mathcal{L}_\mathcal{E}$ is equal to the probability that the starting point in $\mathcal{T}(x)$ was exactly k away from a point already in $\mathcal{L}_\mathcal{E}$, which is negligible since the cardinality of $\mathcal{L}_\mathcal{E}$ is bounded by a polynomial. Hence, the probability that Sx is in $S(\mathsf{Var} - x \cup \mathcal{L}_\mathcal{E}.dom)$ is negligible. □

Proposition 9 (Rule I2)**.** $\{\mathit{Indis}(\nu y; V)\}\ x := y + 1\ \{\mathit{Indis}(\nu x; V)\}$ *if* $x, y \notin V$

Proof. First note that the variable y cannot be in V because the value of x is easily differentiable from a random value when the value of y is given since $x = y + 1$. If there existed an algorithm $\mathcal{A}$ that could differentiate x from a random value given $[\![x := y+1]\!]S(V)$, then the algorithm $\mathcal{B}(S(y), S(V))$ defined by running $\mathcal{A}(S(y)+1, S(V)) = \mathcal{A}([\![x := y+1]\!]S(x), [\![x := y+1]\!]S(V))$ (because $S(V) = [\![x := y + 1]\!]S(V)$ since $x \notin V$). □

Proposition 10 (Rule I3)**.** $\{\mathit{Indis}(\nu t; V)\}\ x := y + 1\ \{\mathit{Indis}(\nu t; V)\}$ *if* $x \notin V$ *even if* $t = y$

Proof. Since $x \notin V$, we have that $S(V) = [\![x := y + 1]\!]S(V)$, so any adversary that can distinguish the value of y from a random value given $[\![x := y \oplus z]\!]S(V)$ could just as easily distinguish the value of y from a random value given $S(V)$. □

Proposition 11 (Rule I4)**.** $\{\mathit{Indis}(\nu t; V, y)\}\ x := y + 1\ \{\mathit{Indis}(\nu t; V, x, y)\}$ *if* $x \notin V$

Proof. This follows from Lemma 3 since $y \in V$ and clearly $y+1$ is constructible from y. □

Proposition 12 (Rule I5)**.** $\{\mathit{Icounter}(y; V_1) \wedge \mathsf{E}(\mathcal{E}; t; V_2)\}\ x := y+1\ \{\mathsf{E}(\mathcal{E}; t; V_2, x)\}$ *even if* $t = y$

Proof. We know from $\mathsf{E}(\mathcal{E}; t; V_2)$ that the probability that the value of t is in $\mathcal{L}_\mathcal{E} \cup V_2$ is negligible. So, to show that $\mathsf{E}(\mathcal{E}; t; V_2, x)$ holds, it suffices to show that the probability that the value of x is equal to the value of t is negligible. The proof is separated in two parts, depending on whether or not $t \in V_1$.

If $t \in V_1$, then since $V_1 = \mathcal{T}(y)$, we have that $S(t) = S(y) - k$ for some non-negative value k that is polynomial in the security parameter. Therefore, clearly, $S(t) \neq S(x) = S(y) + 1$.
If $t \notin V_1$, then by definition of $\mathsf{lcounter}()$, $S(t)$ is indistinguishable from a random value given $S(y)$. This means, among other things, that with high probability, $S(t) \neq S(y) + 1$. Hence, $S(t) \neq S(x)$. □

A.2 Xor operator

Proposition 13 (Rule X1)**.** $\{\mathit{Indis}(\nu y; V, y, z)\}\ x := y \oplus z\ \{\mathit{Indis}(\nu x; V, x, z)\}$ *provided* $y \neq z$ *and* $x, y, z \notin V$

Proof. Let X be a distribution such that $X \models \mathsf{Indis}(\nu y; V, y, z)$, which we can rewrite $X \sim_{(V,y,z)} \nu y.X$. Moreover, $y \oplus z$ is constructible from (V, y, z). We apply lemma 2 to obtain $[\![x := y \oplus z]\!](X) \sim_{V,x,y,z} [\![x := y \oplus z]\!](\nu y.X)$, and by weakening it we get $[\![x := y \oplus z]\!](X) \sim_{V,x,z} [\![x := y \oplus z]\!](\nu y.X)$.

$$
\begin{aligned}
D([\![x := y \oplus z]\!](\nu y.X), V \cup \{x, z\}) &= [S \xleftarrow{\$} X; u \xleftarrow{\$} \mathcal{U}; S' := S\{y \mapsto u\}; \\
&\quad S'' \xleftarrow{\$} [\![x := y \oplus z]\!](S') : S''(V \cup \{x, z\})] \\
&= [S \xleftarrow{\$} X; u \xleftarrow{\$} \mathcal{U}; S' := S\{y \mapsto u\}; \\
&\quad S'' := S'\{x \mapsto u \oplus S(z)\} : S''(V \cup \{x, z\})] \\
&\quad \text{and since xor is idempotent we can write:} \\
&= [S \xleftarrow{\$} X; v \xleftarrow{\$} \mathcal{U}; S'' := S\{x \mapsto v; y \mapsto v \oplus S(z)\} : \\
&\quad S''(V \cup \{x, z\})] \\
&\quad \text{but changing } y \text{ is useless since } y \notin V \cup \{z\} \\
&= [S \xleftarrow{\$} X; v \xleftarrow{\$} \mathcal{U}; S'' := S\{x \mapsto v\} : \\
&\quad S''(V \cup \{x, z\})] \\
&= D(\nu x.X, V \cup \{x, z\})
\end{aligned}
$$

Another way to write this equality of distributions is $[\![x := y \oplus z]\!](\nu y.X) \sim_{V,x,z} \nu x.X$. Then, by transitivity of indistinguishability, we can conclude that $[\![x := y \oplus z]\!](X) \sim_{V,x,z} \nu x.X$, which we can rewrite $[\![x := y \oplus z]\!](X) \models \mathsf{Indis}(\nu x; V, x, z)$. □

Proposition 14 (Rule X2)**.** $\{\mathit{Indis}(\nu t; V)\}\ x := y \oplus z\ \{\mathit{Indis}(\nu t; V)\}$ *if* $x \notin V$*, even if* $t = y$ *or* $t = z$

Proof. Since $x \notin V$, we have that $S(V) = [\![x := y \oplus z]\!]S(V)$, so any adversary that can distinguish the value of y from a random value given $[\![x := y \oplus z]\!]S(V)$ could just as easily distinguish the value of y from a random value given $S(V)$. □

Proposition 15 (Rule X3)**.** $\{\mathit{Indis}(\nu t; V, y, z)\}\ x := y \oplus z\ \{\mathit{Indis}(\nu t; V, x, y, z)\}$

Proof. This follows from Lemma 3 since x is clearly constructible from y and z. □

A.3 Block Cipher

Proposition 16 (Rule B1)**.** $\{\mathsf{E}(\mathcal{E}; y; \emptyset)\}\ x := \mathcal{E}(y)\ \{\mathsf{Indis}(\nu x) \wedge \mathsf{Icounter}(x; \{x\})\}$

Proof. We use the assumption that $\mathcal{E}$ is an ideal cipher, i.e. it is a function sampled at random among all the functions from $\{0,1\}^\eta$ to $\{0,1\}^\eta$. Since, with overwhelming probability, $\mathcal{E}(y)$ has never been computed before, $\mathcal{E}(y)$ looks random and independent from all other values, i.e. until $\mathcal{E}$ is computed at y a second time - which the adversary cannot do on its own - $x := \mathcal{E}(y)$ is indistinguishable from $x \xleftarrow{\$} \mathcal{U}$. Therefore, x is assigned the same invariants as in (R1). □

Proposition 17 (Rule B2)**.** $\{\mathsf{Indis}(\nu t; V) \wedge \mathsf{E}(\mathcal{E}; y; \emptyset)\}\ x := \mathcal{E}(y)\ \{\mathsf{Indis}(\nu t; V, x)\}$ *provided* $\mathcal{L}_\mathcal{E} \notin V$

Proof. As above, since, with overwhelming probability, y has never been queried to the block cipher, then, following the execution of $x := \mathcal{E}(y)$, the value of x is indistinguishable from a random value and independent from all other values of the system, so x cannot be of any help in distinguishing z from random, as in the proof of rule (R2). □

Proposition 18 (Rule B3)**.** $\{\mathsf{Indis}(\nu t; V, \mathcal{L}_\mathcal{E}, y)\}\ x := \mathcal{E}(y)\ \{\mathsf{Indis}(\nu t; V, \mathcal{L}_\mathcal{E}, x, y)\}$

Proof. Note that, as a result of the command $x := \mathcal{E}(y)$, the value of y gets added to the set $\mathcal{L}_\mathcal{E}.dom$, so we need to make sure that t is indistinguishable from random when the value of y in addition to all the values in $\mathcal{L}_\mathcal{E}.dom$ in order to preserve its independence from $\mathcal{L}_\mathcal{E}.dom$. Otherwise, the argument goes exactly as in the proof of rule B2. □

Proposition 19 (Rule B4)**.** $\{\mathsf{Icounter}(t; V)\}\ x := \mathcal{E}(y)\ \{\mathsf{Icounter}(t; V)\}$ *even if* $t = y$

Proof. From the definition of $\mathsf{Icounter}(z)$, we easily see that this is a special case of Rule B2 since the set $\mathcal{T}(t)$ is unchanged by the command $x := \mathcal{E}(y)$. □

Proposition 20 (Rule B5)**.** $\{\mathsf{E}(\mathcal{E}; t; V, y)\}\ x := \mathcal{E}(y)\ \{\mathsf{E}(\mathcal{E}; t; V, y)\}$

Proof. If we have that the probability that either of $S(y)$ or $S(z)$ is in $S(\mathcal{L}_\mathcal{E}).\mathsf{dom}$ is negligible before the execution of $x := \mathcal{E}(y)$, then we have that the probability that $S(z)$ is in $S(\mathcal{L}_\mathcal{E}).\mathsf{dom}$ is negligible after its execution if and only if $S(y) \neq S(z)$. Clearly, if we had that $S(y) = S(z)$, then $\mathsf{Indis}(\nu y; \{z\})$ would not hold, therefore $S(y) \neq S(z)$. □

A.4 Concatenation

Proposition 21 (Rule C1)**.** $\{\mathit{Indis}(\nu y; V, y, z) \wedge \mathit{Indis}(\nu z; V, y, z)\}\ x := y \| z\ \{\mathit{Indis}(\nu x; V, x)\}$ *if* $y, z \notin V$

Proof. $X \models \mathsf{Indis}(\nu z; V, y, z)$ implies $X \sim_{V,y,z} \nu z.X$, so that in turn $\nu y.X \sim_{V,y,z} \nu y.\nu z.X$. But $X \models \mathsf{Indis}(\nu y; V, y, z)$ can be written as $X \sim_{V,y,z} \nu y.X$. Hence, by transitivity we get $X \sim_{V,y,z} \nu y.\nu z.X$. Since $y||z$ is constructible from (V, y, z), we apply lemma 2 to obtain $[\![x := y||z]\!](X) \sim_{V,x,y,z} [\![x := y||z]\!](\nu y.\nu z.X)$, and by weakening we get $[\![x := y||z]\!](X) \sim_{V,x} [\![x := y||z]\!](\nu y.\nu z.X)$. Using the properties of $||$ and that $\{y, z\} \cap V = \emptyset$, we have $D([\![x := y||z]\!](\nu y.\nu z.X), V \cup \{x\}) = D(\nu x.X, V \cup \{x\})$, and hence by transitivity of indistinguishability, $[\![x := y||z]\!](X) \sim_{V,x;} \nu x.X$. □

Proposition 22 (Rule C2)**.** $\{\mathit{Indis}(\nu t; V, y, z)\}\ x := y \| z\ \{\mathit{Indis}(\nu t; V, x, y, z)\}$ *if* $t \neq x, y, z$

Proof. Since $y||z$ is constructible from (V, y, z), we apply corollary 3 to obtain $[\![x := y||z]\!](X) \models \mathsf{Indis}(\nu t; V, x, y, z)$. □

Proposition 23 (Rule C3)**.** $\{\mathit{Indis}(\nu t; V, y, z)\}\ x := y \| z\ \{\mathit{Indis}(\nu t; V, x, y, z)\}$ *if* $t \neq x, y, z$

Proof. Since $x \notin V$, we have that $S(V) = [\![x := y \| z]\!]S(V)$, so any adversary that can distinguish the value of y from a random value given $[\![x := y \| z]\!]S(V)$ could just as easily distinguish the value of y from a random value given $S(V)$. □

A.5 Generic preservation rules

Let $t \neq x, y, z$ and c be either $x \xleftarrow{\$} \mathcal{U}$, $x := y \| z$, $x := y \oplus z$ or $x := y + 1$.

Proposition 24 (Rule G1)**.** $\{\mathit{lcounter}(t; V)\}\ \mathsf{c}\ \{\mathit{lcounter(t;V)}\}$ *if* $y, z \notin V$

Proof. Note that since $y, z \notin V$, then we can see from the semantics that the command c will not alter $\mathcal{T}(t)$. The proof then proceeds exactly as in the proof of Rule (X3). □

Proposition 25 (Rule G2)**.** $\{\mathsf{E}(\mathcal{E}; t; V)\}\ \mathsf{c}\ \{\mathsf{E}(\mathcal{E}; t; V)\}$ *if* $x \notin V$*, even if* $t = y$ *or* $t = z$

Proof. Since $x \notin V$ and c is not a computation of the block cipher, the values in V and $\mathcal{L}_{\mathcal{E}}.dom$ will be unchanged by the command c. Therefore, if the probability that $S(t) \in S(V \cup \mathcal{L}_{\mathcal{E}}.dom)$ was negligible before the execution of the command, it will remain so after its execution. □

$\mathcal{E}_{OFB}(m_1|m_2|m_3, IV|c_1|c_2|c_3)$

var IV, z_1, z_2, z_3;

$IV \xleftarrow{\$} \mathcal{U}$;	$\{\mathsf{Indis}(\nu IV) \wedge \mathsf{E}(\mathcal{E}; IV; \mathsf{Var} - IV)\}$	(R1)(L2)
$z_1 := \mathcal{E}(IV)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var}) \wedge \mathsf{Indis}(\nu z_1)\}$	(B1)(B2)
$c_1 := m_1 \oplus z_1$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var}) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var}^* - z_1)$	(X1)(X3)
	$\wedge\ \mathsf{Indis}(\nu z_1; \mathsf{Var}^* - c_1)\} \wedge \mathsf{E}(\mathcal{E}; z_1; \mathsf{Var} - c_1 - z_1)\}$	(X2)(L2)
$z_2 := \mathcal{E}(z_1)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var}) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1) \wedge \mathsf{Indis}(\nu z_2)\}$	(B1)(B2)
$c_2 := m_2 \oplus z_2$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var}) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1)$	(X3)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var}^* - z_2) \wedge \mathsf{Indis}(\nu z_2; \mathsf{Var}^* - c_2)$	(X1)(X2)
	$\wedge\ \mathsf{E}(\mathcal{E}; z_2; \mathsf{Var} - c_2 - z_2)\}$	(L2)
$z_3 := \mathcal{E}(z_2)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var}) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1)$	(B2)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var} - z_2) \wedge \mathsf{Indis}(\nu z_3)\}$	(B1)
$c_3 := m_3 \oplus z_3$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var}) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - z_1)\}$	(X3)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var} - z_2) \wedge \mathsf{Indis}(\nu c_3; \mathsf{Var}^* - z_3)$	(X1)

Figure 5: Analysis of OFB encryption mode

$\mathcal{E}_{CTR}(m_1|m_2|m_3, IV|c_1|c_2|c_3)$

var $IV, ctr_1, ctr_2, ctr_3, z_1, z_2, z_3$;

$IV \xleftarrow{\$} \mathcal{U}$;	$\{\mathsf{Indis}(\nu IV) \wedge \mathsf{lcounter}(IV; \{IV\})\}$	(R1)
$ctr1 := IV + 1$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var}^* - ctr1)$	(I3)
	$\wedge\ \mathsf{lcounter}(ctr1) \wedge \mathsf{E}(\mathcal{E}; ctr1; \mathsf{Var})\}$	(I1)
$z_1 := \mathcal{E}(ctr1)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - ctr1) \wedge \mathsf{Indis}(\nu z_1)$	(B1)(B2)
	$\wedge\ \mathsf{lcounter}(ctr1; \{IV, ctr1\})\}$	(B4)
$c_1 := m_1 \oplus z_1$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - ctr1) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var}^* - z_1)$	(X1)(X3)
	$\wedge\ \mathsf{lcounter}(ctr1; \{IV, ctr1\})\}$	(G1)
$ctr2 := ctr1 + 1$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - ctr1 - ctr2)$	(I3)
	$\wedge\ \mathsf{Indis}(\nu c_1; \mathsf{Var}^* - z_1) \wedge \mathsf{E}(\mathcal{E}; ctr2; \mathsf{Var})$	(I1)(I4)
	$\wedge\ \mathsf{lcounter}(ctr2; \{IV, ctr1, ctr2\})\}$	
$z_2 := \mathcal{E}(ctr2)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - ctr1 - ctr2)$	(B2)
	$\wedge\ \mathsf{Indis}(\nu c_1; \mathsf{Var}^* - z_1) \wedge \mathsf{Indis}(\nu z_2)$	(B1)
	$\wedge\ \mathsf{lcounter}(ctr2; \{IV, ctr1, ctr2\})\}$	(B4)
$c_2 := m_2 \oplus z_2$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - ctr1 - ctr2)$	(X3)
	$\wedge\ \mathsf{Indis}(\nu c_1; \mathsf{Var}^* - z_1) \wedge \mathsf{Indis}(\nu c_2; \mathsf{Var}^* - z_2)$	(X1)
	$\wedge\ \mathsf{lcounter}(ctr2; \{IV, ctr1, ctr2\})\}$	(G1)
$ctr3 := ctr2 + 1$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - ctr1 - ctr2 - ctr3)$	(I3)
	$\wedge\ \mathsf{Indis}(\nu c_1; \mathsf{Var}^* - z_1) \wedge \mathsf{Indis}(\nu c_2; \mathsf{Var}^* - z_2)$	
	$\wedge\ \mathsf{E}(\mathcal{E}; ctr3; \mathsf{Var})\}$	(I1)
$z_3 := \mathcal{E}(ctr3)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - ctr1 - ctr2 - ctr3)$	(B2)
	$\wedge\ \mathsf{Indis}(\nu c_1; \mathsf{Var}^* - z_1)$	
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var}^* - z_2) \wedge \mathsf{Indis}(\nu z_3; \mathsf{Var})$	(B1)
$c_3 := m_3 \oplus z_3$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - ctr1 - ctr2 - ctr3)$	(X3)
	$\wedge\ \mathsf{Indis}(\nu c_1; \mathsf{Var}^* - z_1)$	
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var}^* - z_2)$	
	$\wedge\ \mathsf{Indis}(\nu c_3; \mathsf{Var}^* - z_3)\}$	(X1)

Figure 6: Analysis of CTR encryption mode

$\mathcal{E}_{PCBC}(m_1|m_2|m_3, IV|c_1|c_2|c_3)$
var $IV, z_1, z_2, z_3, y_2, y_3$;

$IV \xleftarrow{\$} \mathcal{U}$;	$\{\mathsf{Indis}(\nu IV)\}$	(R1)
$z_1 := IV \oplus m_1$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var}^* - z_1) \wedge \mathsf{Indis}(\nu z_1; \mathsf{Var}^* - IV)$	(X1)(X2)
	$\wedge\ \mathsf{E}(\mathcal{E}; z_1; \mathsf{Var} - IV - z_1)\}$	(L2)
$c_1 := \mathcal{E}(z_1)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - z_1) \wedge \mathsf{Indis}(\nu c_1)\}$	(B1)(B2)
$y_2 := c_1 \oplus m_1$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - z_1) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var}^* - y_2)$	(X2)(X3)
	$\wedge\ \mathsf{Indis}(\nu y_2; \mathsf{Var}^* - c_1)\}$	(X1)
$z_2 := y_2 \oplus m_2$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - z_1) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var}^* - y_2 - z_2)$	(X2)(X3)
	$\wedge\ \mathsf{Indis}(\nu z_2; \mathsf{Var}^* - c_1 - y_2) \wedge \mathsf{E}(\mathcal{E}; z_2; \mathsf{Var} - c_1 - y_2 - z_2)\}$	(X1)(L2)
$c_2 := \mathcal{E}(z_2)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - z_1) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - y_2 - z_2)$	(B2)
	$\wedge\ \mathsf{Indis}(\nu c_2)\}$	(B1)
$y_3 := c_2 \oplus m_2$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - z_1) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - y_2 - z_2)$	(X3)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var}^* - y_3) \wedge \mathsf{Indis}(\nu y_3; \mathsf{Var}^* - c_2)\}$	(X1)(X2)
$z_3 := y_3 \oplus m_3$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - z_1) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - y_2 - z_2)$	(X3)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var}^* - y_3 - z_3) \wedge \mathsf{Indis}(\nu z_3; \mathsf{Var}^* - c_2 - y_3)$	(X1)(X2)
	$\wedge\ \mathsf{E}(\mathcal{E}; z_3; \mathsf{Var} - c_2 - y_3 - z_3)$	(L2)
$c_3 := \mathcal{E}(z_3)$;	$\{\mathsf{Indis}(\nu IV; \mathsf{Var} - z_1) \wedge \mathsf{Indis}(\nu c_1; \mathsf{Var} - y_2 - z_2)$	(B2)
	$\wedge\ \mathsf{Indis}(\nu c_2; \mathsf{Var}^* - y_3 - z_3) \wedge \mathsf{Indis}(\nu c_3)\}$	(B1)(B2)

Figure 7: Analysis of PCBC encryption mode