Symbolic and Computational Reasoning About Cryptographic Modes of Operation

Catherine Meadows
Naval Research Laboratory
Center for High Assurance Computer Systems
Washington DC, USA 20375
catherine.meadows@nrl.navy.mil

***Abstract*—In this paper we develop symbolic and computational representations for a class of cryptographic modes of operation, where the symbolic representations are modeled as elements of a term algebra, and we apply them to the analysis of the computational security of the modes. We derive two different conditions on the symbolic representations, a simple one that is sufficient for security, and a more complex one that is both necessary and sufficient, and prove that these properties hold. The problem of deciding computational security then is reduced to the problem of solving certain disunification problems. We also discuss how these results can be extended.**

This paper subsumes the paper "Symbolic Security Criteria for Blockwise Adaptive Secure Modes of Encryption [14]."

I. Introduction

The use of symbolic methods to analyze the security of cryptographic algorithms and protocols has taken a number of forms. When used at a high level of abstraction to model the algorithms as black boxes, it can be used in the automated analysis of complex security protocols. At the other extreme, symbolic methods can be used to formalize and reason about game transformation proofs used in cryptography, enabling machine assisted proofs of cryptographic security. In this paper we explore methods that inhabit a middle ground: the development of symbolic criteria for security of classes of cryptosystems, in which first a class of cryptosystems is defined and represented symbolically, and then criteria on the symbolic representations are developed that guarantee security under the assumption that the cryptographic primitives used are indistinguishable from random. This approach has been proven useful in the automated synthesis of cryptosystems, making it possible to rapidly generate a large number of possible cryptoalgorithms, and then weed out those that fail the criteria, often using an SMT solver. More recently, however, there has also been an interest in applying the symbolic approach to efficient verification algorithms as well, as is done, for example, by McQuoid, Swope, and Rosulek in [13].

In this paper we apply this approach to a subclass of *cryptographic modes of operation*, or more specifically, a class of modes constructed using a combination of randomly generated λ-bit bitstrings, a permutation f of λ-bit blocks to λ-bit blocks that is indistinguishable from a random function, and bitwise exclusive or. Each cryptosystem in this class takes as input a sequence of λ-bit plaintext blocks and returns as output a sequence of λ-bit ciphertext blocks. The schedule by which plaintext blocks are submitted and ciphertext blocks are returned is a parameter of the cryptosystem.

The security property that we are interested in proving is IND\$-CPA security, originally defined by Rogaway in [16]. Roughly speaking, a cryptosystem is IND\$-CPA secure if an adaptive chosen plaintext (CPA) adversary's advantage in distinguishing between a sequence of blocks returned by the cryptosystem and a randomly generated sequence of blocks is a negligible function of the security parameter λ.

Our main results are that we identify a simple, syntactically checkable condition that guarantees IND\$-CPA security, and a somewhat more complex condition that is both necessary and sufficient for it. We begin by describing a symbolic representation of the bitstrings computed by the cryptosystem as members of a term algebra. We then show how a symbolic condition on encrypted blocks computed by the cryptosystem, formulated as a disunification problem subject to a constraint expressed in terms of the schedule for returning blocks, implies IND\$-CPA security. We then leverage off of this condition to obtain a necessary and sufficient condition for IND\$-CPA security to hold, also expressed in terms of constrained disunification problems.

The class of of cryptosystems we study is an extension of that of Malozemoff et al. in [12], which also concentrates on IND\$-CPA security. The main differences between this work and theirs is that in [12] it is required that the ciphertext is not returned until all the plaintext has been received, while in our case the schedule is a parameter of the cryptosystem (so that in particular the blockwise adaptive attacker of Rogaway [16] falls into this class, although we define it as a property of the cryptosystem rather than of the adversary), and that we prove necessary and sufficient conditions for security as well as identifying classes of secure cryptosystems. The class of cryptosystem we study is also similar to those handled by the Linicrypt model proposed by Carmer and Rosulek in [6] and further applied by McQuoid, Swope, and Rosulek in [13]; we discuss this further in Section II.

As an illustration we describe cipher block chaining, which is known be IND\$-CPA secure under the condition that the encryptor waits until it has received all the plaintext blocks from the adversary before sending ciphertext, but violates IND\$-CPA-security if the encryptor sends each block of ciphertext to the adversary immediately after receiving the plaintext block necessary to compute it [9], [16]. This will be used as a running example throughout much of the paper.

Example I.1. Let F be a block cipher, and let $f = F_K$ denote evaluation of f using a key K. In cipher block chaining n plaintext blocks x_1 through x_n are encrypted as follows:

1) $C_0 = r$, where r is a randomly generated block known as an *initialization vector* (IV), and;
2) $C_i = f(x_i \oplus C_{i-1})$ for $i > 0$.

In CBC, the cipher text blocks are not returned by the encryptor until all the plaintext blocks are received. In that case CBC can be shown to be IND\$-CPA-secure. However, it turns out to be insecure if the encryptor follows a different schedule: e.g. if ciphertext blocks are returned as soon as it is possible to compute them. Consider for example the following attack, in which the adversary is able to create two identical ciphertext blocks.

1) Adversary receives r_1
2) Adversary sends $x_1 = 0$
3) Adversary receives $f(0 \oplus r_1) = f(r_1)$
4) Adversary sends $x_2 = r_1 \oplus f(0 \oplus r_1) = r_1 \oplus f(r_1)$
5) Adversary receives $f(r_1 \oplus f(r_1) \oplus f(r_1)) = f(r_1)$.

The adversary can thus cause two ciphertext blocks to be equal with probability 1, allowing it distinguish between the real ciphertext and a random sequence of blocks with overwhelming probability.

The rest of this paper is organized as follows. In Section II we give some background on symbolic verification of cryptosystems and describe related work. In Section III we describe the basic definitions and results in symbolic analysis that are used in this work. In Section IV we describe the symbolic and computational models used and how they are related. In Section V we state and prove the sufficient condition for IND\$-CPA security, and in Section VII we state and prove the more complex necessary and sufficient conditions. In Section VIII we discuss some ways in which the work can be extended. Finally, in Section IX we conclude the paper and discuss some open problems.

II. Background and Related Work

Probably the earliest work on using symbolic methods for cryptographic proofs of security for protocols in the face of adaptive adversarys is that of Backes, Pfitzmann, and Waidner beginning in [2], in which the underlying encryption primitives are defined in a cryptographic library, and the adversary is a full Dolev-Yao adversary that interacts with principals over a network it completely controls. In later work Micciancio and Panjwani [15] prove a computational soundness and completeness theorem for adaptive adversaries that applies to any CPA encryption scheme. However these works used the free equational theory, and so were not useful for algorithms constructed using cryptographic primitives, since this generally requires the modeling of non-trivial equational theories.

Later work has addressed computational soundness and completeness of symbolic security in different equational theories, but most of this has only led to results for weaker, e.g. non-adaptive adversaries. One notable exception is the work of Kremer and Mazaré [10], which addresses the soundness of static equivalence, a symbolic concept used to express indistinguishability properties, in the face of an adaptive adversary. However, this does not allow arbitrary input from the adversary; instead the adversary is restricted to adaptively choosing from a set of protocol executions with which it interacts passively. This is because the soundness results require that the input to the protocol obey certain typing restrictions, which an adversary will not necessarily follow.

There has also been a substantial body of work that focuses explicitly on cryptographic modes of operation. Gagné et al. [7] have developed a Hoare logic for proving semantic security of block cipher modes of encryption, and a program implementing the logic that can be used to automatically prove their security. However, their work concentrates on heuristically driven theorem proving techniques rather than evaluating symbolic security conditions. We also note the work of Bard [3], who considers circumstances under which security for modes of encryption can be reduced to a collision-freeness property. Although [3] does not address symbolic security directly, our approach to deriving criteria for collision-freeness owes much to it. The work of Malezomoff et al. [12] and Hoang et al. [8] is probably the closest to that in this paper. They prove adaptive chosen plaintext security for cryptographic modes of operation based on deterministic block ciphers [12] and authenticated modes of encryption using block ciphers with tweaks [8] by defining a set of symbolic conditions checked on automatically generated modes. These conditions are proved sufficient for security; however no guarantee is given that all secure modes satisfy them.

Another approach to which ours is closely related is Carmer and Rosulek's Linicrypt model [6], a symbolic model that is used to reason about algorithms that involve hash functions or block ciphers along with additive group operations. The security of an algorithm is analyzed by the use of vector spaces over finite fields describing the structure of the cryptosystems. This has potential applications to a number of types of algorithms, including cryptographic modes of operation , one-time lightweight signature schemes, garbled circuits and hash functions. In particular, applications of Linicrypt to garbled circuits have been addressed by Carmer and Rosulek [6] and applications to hash functions by McQuoid et al. [13].

Finally, a discussion of related work would not be complete without a discussion of Unruh's result on the impossibility of computationally sound symbolic exclusive-or [17]. It is a common practice to design symbolic abstractions so that they ignore equations that are satisfied with only negligible probability. However, Unruh points out, this leads to paradoxical behavior: If one picks a subset S of a set D randomly generated strings of identical length, and then checks if $\sum_{s \in S} \oplus s = 0$, the probability of the equation holding is negligible. However, if one is asked if there is *any* subset S of D such that $\sum_{s \in S} \oplus s = 0$ the probability that the answer is "yes" is overwhelming. Unruh uses this information to construct a protocol in which an adversary is given a sequence of random strings and is challenged to find a subset that exclusive-ors to zero. He then gives a strategy by which a probabilistic polynomial-time adversary can find such a subset with overwhelming probability, although the symbolic representations of the bitstrings in the subset do not themselves exclusive-or to zero.

The reason our own work and the other work cited in this section avoids this problem is because it is not focused on finding equalities, but on distinguishing blocks output by a cryptosystem from the output of a random function. In the problems this work focuses on, the ability to make such a distinction can be shown to dependent upon finding an equation that can be satisfied with non-negligible probability. However, Unruh's counterexample shows the necessity of proving this dependence before a symbolic analysis can be assumed to be computationally sound.

III. Symbolic Preliminaries

A *signature* is a finite set of function symbols Σ of different arities. We write $\mathcal{T}_\Sigma(\mathcal{X})$ for the set of all terms constructed using function symbols from Σ and variables from a countable infinite set $\mathcal{X}$. $\mathcal{T}_\Sigma(\mathcal{X})$ is referred to as a *term algebra.* If $\mathbf{T}$ is a subset of $\mathcal{T}_\Sigma(\mathcal{X})$, we write $Sub(\mathbf{T})$ for the set of subterms of elements of $\mathbf{T}$. Thus, if $\mathbf{T}$ is $\{\mathbf{f(g(x),a),h(f(a,b))}\}$, $Sub(\mathbf{T})$ is $\{\mathbf{f(g(x),a),g(x),a,x,h(f(a,b)),f(a,b),b}\}$.

We write $Var(\mathbf{t})$ (respectively $Sym(\mathbf{t})$) for the set of variables (respectively function symbols) present in a term $\mathbf{t}$.

A Σ*-equation* is a pair $\mathbf{t} = \mathbf{t'}$. A set E of Σ-equations induces a congruence relation $=_E$ on terms $\mathbf{t},\mathbf{t'} \in \mathcal{T}_\Sigma(\mathcal{X})$, so that $\mathbf{t} =_E \mathbf{t'}$ if and only if $\mathbf{t}$ can be made equal to $\mathbf{t'}$ via applications of equations from E. An *equational theory* is a pair (Σ, E), where Σ is an order-sorted signature and E a set of Σ-equations. We will refer to a term algebra $\mathcal{T}_\Sigma(\mathcal{X})$ together with an equational theory (Σ, E) as $(\mathcal{T}_\Sigma(\mathcal{X}), E)$. As an example, suppose that $\Sigma = \{\mathbf{e^{-1}/2, e/2, k/0, a/0}\}$, and $E = \{\mathbf{e^{-1}(x, e(x, y)) = y}\}$. Then $\mathbf{e(k, e^{-1}(k, e(k, a))) =_E e(k, a)}$, by setting $\mathbf{x = k}$ and $\mathbf{y = a}$ in the equation $\mathbf{e^{-1}(x, e(x, y)) = y}$.

A *substitution* σ is a mapping from $\mathcal{X}$ to $\mathcal{T}_\Sigma(\mathcal{X})$ that is the identity on all but a finite subset of $\mathcal{X}$ known as the *domain* of σ. Substitutions are homomorphically extended to $\mathcal{T}_\Sigma(\mathcal{X})$. Application of σ to a term t is denoted by σt.

Remark on Notation III.1. We use the common convention that, when σ is the identity on $\mathbf{x}$, we leave the σ off. Thus, instead of writing $\sigma\mathbf{x} = \mathbf{h}(\sigma\mathbf{y}, \sigma\mathbf{z}), \sigma\mathbf{y} = \mathbf{y}, \sigma\mathbf{z} = \mathbf{z}$, we write $\sigma\mathbf{x} = \mathbf{h(y, z)}$.

The composition of two substitutions is $\sigma\theta\mathbf{X} = \sigma(\theta\mathbf{X})$ for $\mathbf{X} \subset Variables$. A substitution σ is an *E-unifier* of a system of equations $\mathbf{S} = \{\ldots, \mathbf{s_i} =^? \mathbf{t_i}\ldots\}$ if $\sigma\mathbf{s_i} =_\mathbf{E} \sigma\mathbf{t_i}$ for every $\mathbf{s_i} =^? \mathbf{t_j} \in \mathbf{S}$. The notation $=^?$ denotes that $\mathbf{s} =^? \mathbf{t}$ is an equation to be solved. As an example, consider the equation $\mathbf{S} = \{\mathbf{w} =^? \mathbf{e^{-1}(k, z)}\}$ in the algebra $(\mathcal{T}_\Sigma(\mathcal{X}), \{\mathbf{e^{-1}(x, e(x, y)) = y}\})$ described in the previous paragraph. Then the substitutions $\sigma_\mathbf{1} : \mathbf{w} \mapsto \mathbf{e^{-1}(k, z)}$ and $\sigma_\mathbf{2} : \mathbf{z} \mapsto \mathbf{e(k, w)}$ are both unifiers of $\mathbf{S}$ modulo $\mathbf{E}$.

We will be interested in the algebra consisting of a number of *free symbols* that obey no equational theory, plus an exclusive-or operator $\oplus$ and a null operator $\mathbf{0}$. The $\oplus$ operator is associative and commutative and satisfies $\mathbf{X} \oplus \mathbf{0} = \mathbf{0}$ and $\mathbf{X} \oplus \mathbf{X} = \mathbf{0}$. Equality of two terms modulo this theory is equivalent to equality under the theory $(R_\oplus \uplus AC)$ where AC is the associative and commutative rules for $\oplus$, and $R_\oplus$ is a set of *rewrite rules*, $\{\mathbf{X} \oplus \mathbf{0} \to \mathbf{X},\ \mathbf{X} \oplus \mathbf{X} \to\ \mathbf{0}\}$ oriented from left to right. A rewrite rule $\ell \to \mathbf{r}$ is applied to a term $\mathbf{t}$ by finding a subterm $\mathbf{s}$ of $\mathbf{t}$ such that $\mathbf{s} = \sigma\ell$ modulo AC for some substitution σ, and replacing $\mathbf{s}$ in $\mathbf{t}$ with $\sigma\ell$. Thus $\mathbf{0 \oplus a \oplus b}$ can be reduced to $\mathbf{a \oplus b}$ by noting that $\mathbf{0 \oplus a \oplus b} = (\mathbf{0 \oplus a}) \oplus \mathbf{b} = \sigma\ell \oplus \mathbf{b}$ modulo AC, where ℓ is the left-hand side of $\mathbf{X} \oplus \mathbf{0} \to \mathbf{X}$, and $\sigma\mathbf{X} = \mathbf{a}$. In addition, every term $\mathbf{t}$ reduces after a finite number of steps to a *normal form* $\downarrow_{\mathbf{R}_\oplus,\mathbf{AC}}\mathbf{t}$ to which no further rewrite rules can be applied, and this normal form is unique up to AC-equivalence. We refer to the theory $(\mathbf{R}_\oplus \uplus \mathbf{AC})$ as the $\oplus$ theory for brevity.

IV. Symbolic and Computational Models

In this section we give a brief description of how we model symbolic terms and computational functions. This is based on the abstract and concrete models of cryptosystems introduced by Baudet et al. [4], with the main difference that in some cases we allow variables in the symbolic model to be replaced in the computational model by probabilistic Turing machines, instead of restricting ourselves to concrete computational representations of symbolic terms.

Remark on Notation IV.1. Note that we write symbolic terms in **boldface**, and computational functions in *italic* in order to make it easier to distinguish between them.

Since IND\$-CPA security is defined in terms of messages that are all encrypted with the same key, we can represent the block cipher function $F_K(\_)$, where K is the key used by the encryptor, simply as $f(\_)$. In addition IND\$-CPA security requires the assumption that the output of the block cipher be indistinguishable from random by the adversary. Thus, in the remainder of this paper, since we will not have a need to actually invert f, we will assume that the encryption function f is actually random.

Let f be a random function from $\{0,1\}^\lambda$ to $\{0,1\}^\lambda$; that is, one in which for each $x \in \{0,1\}^\lambda$, $f(x)$ is chosen uniformly at random from $\{0,1\}^\lambda$. Let Σ be a set of symbols $\{\mathbf{f}, \oplus, \mathbf{0}\}$ and let $\oplus$ be the exclusive-or theory, as described in Section III. We consider the cryptographic term algebra $\mathcal{T}_{(\Sigma,\oplus)}(\mathcal{X})$ where $\mathcal{X}$ is a countable set of variables. There are two types of variables in $\mathcal{X}$: bound variables standing for a string chosen uniformly at random from $\{0,1\}^\lambda$, that are quantified by ν, and free variables with no quantifiers. Generally, quantifiers are specified at the beginning of an expression. So if $\nu\mathbf{r_1}\ldots\nu\mathbf{r_n}.\mathbf{T}$ is an expression (for example a list of terms), the appearance of the quantified variable $\nu\mathbf{r_i}$ before $\mathbf{T}$ means that $\mathbf{r_i}$ stands for the same randomly chosen string wherever it appears in $\mathbf{T}$.

We may omit the quantifiers when referring to $\nu\mathbf{r_1}\ldots\nu\mathbf{r_n}.\mathbf{T}$ when confusion can be avoided. We refer to $\mathcal{T}_{(\Sigma,\oplus)}(\mathcal{X})$ as the $MOO_\oplus$ algebra (for Mode Of Operation), and terms from $\mathcal{T}_{(\Sigma,\oplus)}(\mathcal{X})$ as $MOO_\oplus$ terms.

Except for the fact that the number of bound variables is infinite, they are treated as constants in the term algebra. In particular, no substitutions can be made to a bound variable. Thus, we consider a term *ground* if it contains only function symbols and bound variables.

We say that a term $\mathbf{t}$ is $\mathbf{g}$-*rooted* if it is of the form $\mathbf{g}(\mathbf{s_1},\ldots,\mathbf{s_k})$ where $\mathbf{s_1},\ldots,\mathbf{s_k}$ are terms and $\mathbf{g}$ is a function symbol. We say that a term is *random* if it is either $\mathbf{f}$-rooted or a bound variable.

Let λ be a security parameter. Each sequence of n terms $\mathbf{T} \subset \mathcal{T}_{(\Sigma,\oplus)(\mathcal{X})}$ containing no free variables determines a probability distribution $\mathbf{T}$ over $\{0,1\}^{\lambda\cdot n}$. called the *computational realization T of* $\mathbf{T}$. (Note that, as is described in Remark IV.1, the computational realization T of $\mathbf{T}$ is written in *italics* to distinguish it from the symbolic $\mathbf{T}$, which is written in **bold**.) This is defined as follows: To compute the output of a bound variable $\mathbf{r}$, we choose a λ-length bitstring uniformly at random. If $\mathbf{r}$ occurs more than once in $\mathbf{T}$, it is replaced with the same random λ-length bitstring wherever it occurs. We also replace 0 by a bitstring of λ zeroes, $\oplus$ by bitwise exclusive-or on λ-length bitstrings, and f by a random function.

We can use this recipe for computing a sequence of n λ-length bitstrings we will call the *output of* T or $[\![T]\!]_\lambda$. When we can do so without confusion, we will drop the λ. Note that, if $\mathbf{T}$ contains any bound variables, then T can have more than one possible output, based which on which bitstrings are chosen for the outputs of the bound variables, and each output will be produced with a certain probability.

We now consider the case where $\mathbf{T}$ contains free variables. Free variables play a special role: they are place holders for inputs to a term. In our case they will always stand for inputs from the adversary. In particular, the free variables $\mathbf{x}_1,\ldots\mathbf{x}_\ell$ appearing in $\mathbf{T}$ stand for inputs supplied to the computational realization T of $\mathbf{T}$ by $tm_1,\ldots,tm_\ell$ where each tm_i represents a probabilistic Turing machine. Note that we do not require that the tm_i themselves be computational realizations of elements of $\mathcal{T}_{(\Sigma,\oplus)}(\mathcal{X})$; they can be arbitrary probabilistic Turing machines. We call a map θ that maps a finite set of variables of $\mathcal{T}_{(\Sigma,\oplus)}(\mathcal{X})$ to probabilistic Turing machines and is the identity on all other variables a *computational substitution*. If θ is a computational substitution, and $\mathbf{T}$ is a finite subset of $\mathcal{T}_{(\Sigma,\oplus)}(\mathcal{X})$, we define θT to be the substitution obtained by replacing each variable x used by the computational functions in T with θx. The composition of two computational substitutions is defined in the natural way.

If a θ is a substitution so that θt contains no free variables, we define $[\![\theta t]\!]$ to be the output obtained from t by using the output $[\![\theta x]\!]$ wherever a free variable x appears.

V. $MOO_\oplus$ Cryptosystems and Symbolic Histories

We make the assumption that the cryptosystems we are studying are generated by a program of bounded size whose running time is bounded by a polynomial function of the size of its input, and that both the adversary and the encryptor are uniform probabilistic polynomial time Turing machines.

A cryptographic mode of operation may be thought of as a probabilistic mapping from a sequence of plaintext blocks to a sequence of ciphertext blocks, together with a schedule for delivering the ciphertext blocks (e.g., as soon as possible, or not until all plaintext blocks have been received). Modes are generally defined recursively. But for the purpose of proving results about symbolic analysis, it will be more convenient for us to think of a mode in terms of its possible histories, even if this is not the most compact way of representing it.

A *symbolic $MOO_\oplus$ history* (symbolic history for short) describes the messages exchanged during a process in which an adversary interacts with the oracle to encrypt a message, where the encrypted message is a sequence of $MOO_\oplus$ terms. Histories are analogous to the *frames* defined by Abadi and Fournet in [1]; that is, they represent a record of a protocol execution.[1] Histories are presented in terms of sequences of events in which computations and messages sent and received are represented by elements of $\mathcal{T}_{(\Sigma,\oplus)}(\mathcal{X})$. A symbolic history may contain free variables, that stand for possible data that could be sent or computed by the principals, and variables bound by the quantifier ν, that stand for uniformly randomly generated bitstrings. In the rest of this section we describe in more detail how symbolic histories are created and used.

A. Using Symbolic Histories to Specify Encryption of a Sequence of Plaintext Blocks

The encryption of a message consisting of a sequence of plaintext blocks can be thought of as a protocol described symbolically as a sequences of exchanges between an adversary and an encryptor. In each step the adversary submits one or zero plaintext blocks to the encryptor, where each plaintext block is represented by a unique free variable. The adversary also gives instructions on what functions are to be computed on its input, with the restriction that these functions must be consistent with the definition of the cryptosystem. The encryptor then computes the output of a (possibly empty) set of computational realizations of $MOO_\oplus$ terms, using the plaintext blocks previously sent by the adversary to replace any free variables appearing in the terms. We can thus represent the communication between the adversary and the encryptor at the symbolic level by a symbolic history of the form

$$\nu\mathbf{R}[\mathbf{I_1},\mathbf{O_1},\ldots,\mathbf{I_k},\mathbf{O_k}]$$

where each $\mathbf{I_j}$ is either the empty sequence or a unique free variable, and each $\mathbf{O_j}$ is a (possibly empty) sequence of $MOO_\oplus$ terms whose only free variables are elements of $\{\mathbf{I_j} | 1 \le i \le j\}$, and $\mathbf{R}$ is the set of bound variables appearing in the history.

[1]We could indeed define histories as frames, but because in this work we don't have any need of the main feature of frames (that they are substitutions) we choose a simpler option.

The adversary may encrypt multiple messages and even interleave such encryptions. These activities can also be represented by symbolic histories.

To give an example of a symbolic history, we consider the case, using cipher block chaining, in which an adversary interacts with an encryptor that allows it to encrypt two messages in parallel, timing its response so that that the adversary receives the two IVs first, and the two first blocks of ciphertext right after sending the two first blocks of plaintext:

$$\nu\mathbf{r_1}.\nu\mathbf{r_2}[\emptyset, \emptyset, \mathbf{r_1}, \mathbf{r_2}, \mathbf{x_1}, \mathbf{x_2}, \mathbf{f}(\mathbf{r_1} \oplus \mathbf{x_1}), \mathbf{f}(\mathbf{r_2} \oplus \mathbf{x_2})]$$

VI. $MOO_{\oplus}$ Games and a Simple Criterion for Security

In this section we describe games involving an adversary and an encryptor that together construct a symbolic history and its output. These games will be used both to define and prove security properties of $MOO_{\oplus}$ cryptosystems.

In the following, we use a slight abuse of language that helps us to avoid awkwardness when talking about substitutions used in a game. Normally, the functions computed by an adversary are not represented explicitly in a cryptographic game. Rather, the adversary *is* the function. However, we need to represent a substitution σ to a variable computed by an adversary explicitly as a map from a variable x to σx. We will use the convention that any such function σ is the function that the adversary is programmed to compute. In other words, it is not choosing an arbitrary function.

A. Definition of the Games Used in the Proof

We begin by describing the IND\$-CPA game between the adversary and the real and random encryptors as follows. We assume that the adversary is a uniform probabilistic polynomial Turing machine and that both real and random encryptors are polynomially bounded. In addition, we assume that the adversary cannot compute the function f (or its inverse) itself.

The game proceeds in a series of steps. In each step the adversary sends the encryptor instructions for what it wants computed along with a (possibly empty) sequence of plaintext blocks. Irrespective of whether it is real or random, the encryptor checks that the request is valid according to the definition of the cryptosystem. If it is not, the game is aborted. If the request is valid, the real encryptor returns the encrypted blocks specified by the cryptosystem, at the specified times. The random encryptor returns the same number of blocks at the same times as the real encryptor, but the blocks are independently and uniformly randomly chosen. At any time the adversary may halt the game and guesses whether it is interacting with the real or random encryptor, outputting 1 if it guesses the real encryptor, and 0 if it guesses the random one. We say that $\mathcal{A}^f(1^\lambda) = 1$ if the adversary outputs 1 when interacting with the real encryptor, and $\mathcal{A}^\$(1^\lambda) = 1$ if the adversary outputs 1 when interacting with the random encryptor.

Definition VI.1. Let $\mathcal{C}$ be a $MOO_{\oplus}$ cryptosystem. We say that $\mathcal{C}$ is IND\$-CPA secure if the following is a negligible function of the security parameter λ for all probabilistic polynomial-time programs $\mathcal{A}$:

$$|Pr(\mathcal{A}^f(1^\lambda) = 1) - Pr(\mathcal{A}^\$(1^\lambda) = 1)|$$

If we attempt to prove IND\$-CPA security directly, however, we run into complications. For example, consider the case in which the encryptor sends the strings $[\![f(f(a))]\!]$, $[\![f(f(b))]\!]$, and $[\![f(b)]\!]$ to the adversary, who observes that $[\![f(f(a))]\!] \neq [\![f(f(b))]\!]$. This leaks a little bit of information about $[\![f(a)]\!]$, namely that it is not equal to $[\![f(b)]\!]$. Thus, if the the adversary later needs to solve the equation $x = [\![f(a)]\!]$, the probability of its guessing correctly is not $1/2^\lambda$ but at least $1/(2^\lambda - 1)$. In order to get an actual upper bound, it appears that we would need to at least get an estimate of the number of leakages of this kind.

Instead of doing that, we construct a simpler game in which proving IND\$-CPA security is easier, and prove that the original IND\$-CPA game is indistinguishable from the simpler game, using an "identical-until-bad" argument. The only difference between the two games is the way in which f is computed. In the original game, which we will call G_{str}, f is a random function from bitstrings to bitstrings. In the second game, which we will call G_{symb}, f is a function from symbolic terms to bitstrings.

Before we can work with these two games, we need to give more information about how the encryptors in G_{str} and G_{symb} work. In the following, we say "the encryptor" when both encryptors behave the same way. Otherwise we identify the encryptor as a G_{str} or a G_{symb} encryptor.

The encryptor maintains a symbolic history $\mathbf{H}$ describing its interaction so far with the adversary, as well as two databases that describe the output $[\![\sigma H]\!]$. The first database, DBI, stores the plaintext blocks sent by the adversary and the second, DBO, stores results of the random functions computed by the encryptors.

DBO contains two types of tuples. The first are of the form $[instr, \mathbf{F}, outstr]$ where $\mathbf{F}$ is a set of symbolic $\mathbf{f}$-rooted terms. Thus, if $\mathbf{f}(\mathbf{s})$ is an element of $\mathbf{F}$, $instr$ denotes $[\![\sigma s]\!]$, the input of f, and $outstr$ denotes the string returned by $f(instr)$. The second are of the form $[\mathbf{r_i}, [\![r_i]\!]]$, where $\mathbf{r_i}$ is a bound variable and $[\![r_i]\!]$ is a bitstring chosen uniformly at random. The database DBI consists of tuples of the form $[\mathbf{x}, [\![\sigma x]\!]]$, where $\mathbf{x}$ is a free variable, and $[\![\sigma x]\!]$ is the output of σx, where σ is the substitution computed by the adversary. We also assume that at the beginning, $\mathbf{H}$, DBO and DBI are all empty. They are extended by the encryptor as the protocol evolves.

We now describe the computations made by the encryptors. Suppose that an encryptor, whether real or random, receives an input $[\![\sigma I]\!]$ from the adversary, and it is required to return $[\![\sigma O]\!]$. It performs the following steps for each variable or $\mathbf{f}$-rooted term $\mathbf{t} = \mathbf{f}(\mathbf{s}) \in Sub(\mathbf{O})$, that has not been computed already, computing each bitstring $[\![\sigma s]\!]$ such that $\mathbf{s}$ is a proper subterm of $\mathbf{t}$ before computing $[\![\sigma t]\!]$.

1) If $\mathbf{I} = \mathbf{x}$, where $\mathbf{x}$ is a free variable, it stores $[\mathbf{x}, [\![\sigma x]\!]]$ in DBI. If $\mathbf{I}$ is empty, it skips this step.
2) For any bound variable $\mathbf{r} \in Sub(\mathbf{O})$ such that there is no tuple $[\mathbf{r}, str]$ in DBO, it choses a λ-length bitstring str' uniformly at random and stores it as $[\mathbf{r}, str']$ in DBO.
3) For each $\mathbf{f}(\mathbf{t}) \in Sub(\mathbf{O})$ such that there is not already a tuple $[a, \mathbf{W}, b]$ in DBO with $\mathbf{f}(\mathbf{t}) \in \mathbf{W}$, the encryptor computes $instr$ using the outputs stored in DBI and DBO. That is, if

$$\mathbf{t} = (\sum_{\mathbf{i=0}}^{\mathbf{m}} \oplus \alpha_{\mathbf{i}} \mathbf{x_i}) \oplus (\sum_{\mathbf{j=1}}^{\mathbf{n}} \oplus \beta_{\mathbf{i}} \mathbf{r_i}) \oplus (\sum_{\mathbf{j=1}}^{\mathbf{q}} \oplus \gamma_{\mathbf{i}} \mathbf{f}(\mathbf{s_i}))$$

then for each $\alpha_{\mathbf{i}} \neq \mathbf{0}$ (respectively, $\beta_{\mathbf{i}} \neq \mathbf{0}$, $\gamma_{\mathbf{i}} \neq \mathbf{0}$ the encryptor finds $[\mathbf{x_i}, [\![\sigma x_i]\!]] \in DBI$ (respectively, $[\mathbf{r_i}, [\![r_i]\!]] \in DBO$, $[instr, \mathbf{W}, outstr] \in DBO$ such that $\mathbf{f}(\mathbf{s_i}) \in \mathbf{W}$), and computes the exclusive-or sum of the final bitstrings of each tuple found.
4) We now reach the place where G_{str} and G_{symb} differ.
   a) If there is already a tuple $[instr, \mathbf{F}, outstr] \in DBO$, the G_{str} encryptor replaces it with $[instr, \mathbf{F} \cup \{\mathbf{f}(\mathbf{t})\}, outstr]$. Otherwise, it picks a random λ-length bitstring $outstr'$ and stores $[instr, \{\mathbf{f}(\mathbf{t})\}, outstr']$ in DBO.
   b) The G_{symb} encryptor always picks a random λ-length bitstring $outstr$ and stores $[instr, \{\mathbf{f}(\mathbf{t})\}, outstr]$ in DBO.
5) The encryptor uses the output values stored in DBI and DBO to construct $[\![\sigma O]\!]$, and returns it to the adversary.

Example VI.1. We can model the attack described in Example I.1 using the G_{str} encryptor as follows.

1) The adversary sends the encryptor a request for the initialization vector $\mathbf{r_1}$. The encryptor sets $\mathbf{H}$ equal to $\mathbf{r_1}$. The encryptor computes a random bitstring $outstr_0$ and stores $[\{\mathbf{r_1}\}, outstr_0]$ in DBO, and sends $outstr_0$ to the adversary.
2) The adversary computes $[\![\sigma x_1]\!] = 0^\lambda$ and sends it to the encryptor, along with

   $$\{\mathbf{x_1}\}\{\mathbf{f}(\mathbf{x_1} \oplus \mathbf{r_1})\}$$

   The encryptor sets $\mathbf{H} = \mathbf{r_1}.\mathbf{x_1}.\mathbf{f}(\mathbf{x_1} \oplus \mathbf{r_1})$. It then stores $[\mathbf{x_1}, 0^\lambda]$ in DBI. It then chooses a random bitstring $outstr_1$ and sets

   $$[\![\sigma_1 f(x_1 \oplus r_1)]\!] = [\![f(0 \oplus r_1)]\!] = f(outstr_0) = outstr_1$$

   and stores $[outstr_0, \{\mathbf{f}(\mathbf{x_1} \oplus \mathbf{r_1})\}, outstr_1]$ in DBO. Finally, it returns $[\![f(r_1)]\!] = outstr_1$ to the adversary.
3) The adversary computes $[\![\sigma x_2]\!] = [\![r_1]\!] \oplus [\![f(r_1)]\!] = outstr_0 \oplus outstr_1$ and sends it to the encryptor, along with the sequence $\mathbf{x_2}.\mathbf{f}(\mathbf{x_2} \oplus \mathbf{f}(\mathbf{x_1} \oplus \mathbf{r_1}))$. $\mathbf{H}$ is updated by the encryptor as before. The encryptor computes $[\![\sigma(x_2 \oplus f(x_1 \oplus r_1))]\!] = [\![r_1]\!] \oplus [\![f(r_1)]\!] \oplus [\![f(r_1)]\!] = outstr_0 \oplus outstr_1 \oplus outstr_1 \oplus outstr_0$. It finds the tuple $[outstr_0, \{\mathbf{f}(\mathbf{x_1} \oplus \mathbf{r_1})\}, outstr_1]$ in DBO, which it replaces with $[outstr_0, \{\mathbf{f}(\mathbf{x_1} \oplus \mathbf{r_1}), \mathbf{f}(\mathbf{x_2} \oplus \mathbf{f}(\mathbf{x_1} \oplus \mathbf{r_1}))\}, outstr_1]$. It then sends $outstr_1$ to the adversary.

We note that such an attack is *not* possible when the G_{symb} encryptor is used. In that case the strings returned by the encryptor will be independently randomly generated, since the symbolic terms $\mathbf{r_1}, \mathbf{f}(\mathbf{x_1} \oplus \mathbf{r_1})$ and $\mathbf{f}(\mathbf{x_2} \oplus \mathbf{f}(\mathbf{x_1} \oplus \mathbf{r_1})$ are all different. Indeed, as we shall see, the only cases in which the strings returned by the G_{symb} encryptor are *not* independent is the case in which there is a symbolic history $\mathbf{H}$ with a subsequence $\mathbf{D}$ such that the exclusive-or sum of the elements of $\mathbf{D}$ is 0. Thus, when we want to prove that the output of a G_{str} encryptor is indistinguishable from random, it will be enough, except in certain trivially insecure cases, to show that it is indistinguishable from the output of the corresponding G_{symb} encryptor.

We note that the G_{str} encryptor, when computing the output of a term $\mathbf{f}(\mathbf{t})$, chooses a new output string uniformly at random if and only if $[\![\sigma t]\!]$ is a new bitstring input for f, while the G_{symb} encryptor chooses a new output string uniformly at random if and only $\mathbf{f}(\mathbf{t})$ is a new symbolic term. Thus the function f used by the G_{str} encryptor is a random function from bitstrings to bitstrings, while the f used by the f_{symb} is a function from symbolic terms to bitstrings. Moreover, their behavior only differs in the case that $[\![\sigma s]\!] = [\![\sigma t]\!]$ but $\mathbf{s} \neq \mathbf{t}$. We can thus write "identical-until-bad" programs for f_{str} and f_{symb}, shown in pseudocode in Algorithm 1, where the underlined code after *bad* is set to 1 is executed by the G_{str} encryptor.

Algorithm 1 Algorithm for G_{str} and $\underline{G_{symb}}$ computation of f

```
1: if $\mathbf{t} = \mathbf{f}(\mathbf{s})$ and there is no $[a, \mathbf{U}, b] \in DBO$ such that $\mathbf{f}(\mathbf{s}) \in \mathbf{U}$ then
2:     $\mathbf{W} \leftarrow \emptyset$
3:     Construct $instr = [\![\sigma s]\!]$ from $DBO$ and $DBI$
4:     $outstr \leftarrow\$ \{0,1\}^\lambda$
5:     if $\exists \mathbf{F}, outstr1\ s.t.\ [instr, \mathbf{F}, outstr1] \in DBO$ then
6:         $bad \leftarrow 1$
7:         $\underline{outstr \leftarrow outstr1}$
8:         $\underline{\mathbf{W} \leftarrow \mathbf{F}}$
9:         $\underline{DBO \leftarrow (DBO \setminus \{[instr, \mathbf{F}, outstr1]\})}$
10:    end if
11:    $DBO \leftarrow DBO \cup \{[instr, \mathbf{W} \cup \{\mathbf{f}(\mathbf{s})\}, outstr]\}$
12: end if
```

Thus, by the fundamental lemma of game-playing [5], the probability of the adversary's being able to distinguish between the use of the f_{str} and f_{symb} functions in constructing $\mathbf{H}$ is bounded by the probability that *bad* is set to 1 in either of them. This only happens when $[\![\sigma u]\!] = [\![\sigma v]\!]$, where $\mathbf{f}(\mathbf{u}), \mathbf{f}(\mathbf{v})$ are subterms of terms returned by the encryptor, so we will concentrate on estimating the probability of this occurring when the f_{symb} function is used.

B. Symbolic Criterion and Proof of its Sufficiency of IND$-CPA *Security*

We first give some definitions that will allow us to define the symbolic condition. We begin by defining a canonical form for elements of $Sub(\mathbf{H})$.

Definition VI.2. Let $\mathbf{H}$ be a symbolic history, and suppose that $\mathbf{v}$ is the exclusive-or of non-$\oplus$-rooted elements of $Sub(\mathbf{H})$ We say that $\mathbf{v}$ is in $\mathbf{H}$-*canonical form* if

$$\mathbf{v} = (\sum_{i=0}^{\ell} \oplus \alpha_i \mathbf{x_i}) \oplus (\sum_{i=0}^{k} \oplus \beta_i \mathbf{g_i})$$

where:

1) Each $\mathbf{g_j}$ is either a bound variable or $\mathbf{f}$-rooted term from $Sub(\mathbf{H})$.
2) Each $\mathbf{x}_i$ is the ith free variable chosen by the adversary.
3) Each α_i (respectively β_i) is either 0 or 1, and if at least one $\alpha_i = 1$ then $\alpha_\ell = 1$. In that case we call $\mathbf{x}_\ell$ the *leading free variable of* $\mathbf{v}$.

If the $\mathbf{H}$-canonical form of $\mathbf{v}$ has a leading free variable x_ℓ, we call

$$\sigma \mathbf{x}_\ell = (\sum_{i=0}^{\ell-1} \oplus \alpha_i \mathbf{x_i}) \oplus (\sum_{i=0}^{k} \oplus \beta_i \mathbf{g_i})$$

the $\mathbf{H}$-*ordered substitution derived from* $\mathbf{v}$. (Recall from Remark III.1 that, that this notation means that σ is the identity on $\mathbf{x_1}$ through $\mathbf{x_{\ell-1}}$.)

We often denote the $\mathbf{H}$-canonical form of a term as $\mathbf{X} \oplus \mathbf{G}$, where $\mathbf{X}$ is either zero or the sum of free variables, and $\mathbf{G}$ is either zero or the sum of bound variables and $\mathbf{f}$-rooted terms. If a term is known to have a leading free variable $\mathbf{x}_\ell$, we may also write its $\mathbf{H}$-canonical form as $\mathbf{x}_\ell \oplus \mathbf{X} \oplus \mathbf{G}$.

Now, we define a relation between free variables and $\oplus$ sums of elements of R_H that will tell us what kind of substitutions an adversary can make to the free variables in $\mathbf{H}$.

Definition VI.3. Let $\mathbf{H}$ be a symbolic history $\nu\mathbf{R}[\mathbf{I_1}.\mathbf{O_1}.\ldots.\mathbf{I_k}.\mathbf{O_k}]$. We define $<_{\mathbf{H}}$ between sums $\mathbf{G}$ of bound variables and $\mathbf{f}$-rooted terms appearing in $\mathbf{H}$, and the free variables $\mathbf{x}$ appearing in $\mathbf{H}$ as follows. First, let ℓ be the integer such that $\mathbf{I}_\ell = \{\mathbf{x}\}$. We say $\mathbf{G} <_{\mathbf{H}} \mathbf{x}$ if and only if either $\mathbf{G}$ = 0, or $\mathbf{G} = \oplus_{\mathbf{i=1}}^{\mathbf{m}} \mathbf{G_i}$ such that for each $\mathbf{G_i}$, either $\mathbf{G_i} \in \mathbf{O_j}$ for some $j < \ell$, or there is an $\mathbf{X_i}$ such that $\mathbf{G_i} \oplus \mathbf{X_i} \in \mathbf{O_j}$ for some $j < \ell$;

In other words, $\mathbf{G} <_{\mathbf{H}} \mathbf{x}$ if and only the adversary is able to compute $[\![\sigma G]\!]$ from information it has received from the encryptor before computing and sending $[\![\sigma x]\!]$.

Now, we are able to define safe and unsafe:

Definition VI.4. Let $\mathbf{H}$ be a symbolic history, and let $\mathbf{f}(\mathbf{u_1})$ and $\mathbf{f}(\mathbf{u_2})$ be two different $\mathbf{f}$-rooted elements of $Sub(\mathbf{H})$.We say that $(\mathbf{f}(\mathbf{u_1}), \mathbf{f}(\mathbf{u_2}))$ is *unsafe with respect to* $\mathbf{H}$ if the $\mathbf{H}$-canonical from of $\mathbf{u_1} \oplus \mathbf{u_2}$ is $\mathbf{x}_\ell \oplus \mathbf{X} \oplus \mathbf{G}$, where either $\mathbf{G} = \mathbf{0}$ or $\mathbf{G} <_{\mathbf{H}} \mathbf{x}_\ell$, and that is is *safe* otherwise.

We say that a symbolic history $\mathbf{H}$ is *unsafe* if there are two terms $\mathbf{f}(\mathbf{u_1})$ and $\mathbf{f}(\mathbf{u_2})$ in $\mathbf{H}$ such that $(\mathbf{f}(\mathbf{u_1}), \mathbf{f}(\mathbf{u_2}))$ is unsafe with respect to $\mathbf{H}$, and that it is safe otherwise. We say that a $MOO_\oplus$ cryptosystem is *unsafe* if it is possible to construct an unsafe symbolic history for it, and that it is safe otherwise.

In addition we say that an $\mathbf{H}$-ordered substitution $\sigma \mathbf{x}_\ell = X \oplus \mathbf{G}$ is *computable* if $\mathbf{G} <_{\mathbf{H}} \mathbf{x}_\ell$, and non-computable otherwise, and that equation $\mathbf{u_1} =_\oplus \mathbf{v_1}$ is *solvable* if $(\mathbf{f}(\mathbf{u_1}), \mathbf{f}(\mathbf{u_2}))$ is unsafe.

Example VI.2. Consider blockwise CBC as shown in Example I.1. Consider the symbolic history

$$\mathbf{H} = \nu\mathbf{r}[\emptyset.\mathbf{r}.\mathbf{x_1}.\mathbf{f}(\mathbf{x_1} \oplus \mathbf{r}).\mathbf{x_2}.\mathbf{f}(\mathbf{x_2} \oplus \mathbf{f}(\mathbf{x_1} \oplus \mathbf{r}))]$$

The $\mathbf{H}$-canonical form of the sum of the arguments of the two f-rooted terms in $\mathbf{H}$ is $\mathbf{x_1} \oplus \mathbf{x_2} \oplus \mathbf{r} \oplus \mathbf{f}(\mathbf{x_1} \oplus \mathbf{r})$ with $\mathbf{x_2}$ the leading free variable. We have $\mathbf{x_2} >_{\mathbf{H}} \mathbf{r}$ and $\mathbf{x_2} >_{\mathbf{H}} \mathbf{f}(\mathbf{x_1} \oplus \mathbf{r})$, and so $\mathbf{H}$ is unsafe.

Before we give the main theorem of this section, we state and prove the following straightforward consequence of unsafeness.

Proposition VI.1. *Let* $\mathbf{H}$ *be an unsafe symbolic history, and let* $(\mathbf{f}(\mathbf{u_1}), \mathbf{f}(\mathbf{u_2}))$ *be an unsafe pair with respect to* $\mathbf{H}$. *There is an adversary that can compute* σ *such that* $[\![\sigma u_1]\!] = [\![\sigma u_2]\!]$ *with probability 1, in both* G_{str} *and* G_{symb}.

Proof. Let $\mathbf{X} \oplus \mathbf{G}$ be the $\mathbf{H}$-canonical form of $\mathbf{u_1} \oplus \mathbf{u_2}$. If $\mathbf{G} = \mathbf{0}$ then the adversary can solve $[\![\sigma u_1]\!] = [\![\sigma u_2]\!]$ by setting $[\![\sigma x]\!] = 0$ for all $\mathbf{x} \in \mathbf{X}$. Otherwise, suppose $\mathbf{G} \neq \mathbf{0}$. Then we can write the $\mathbf{H}$-ordered canonical form as $\mathbf{x}_\ell \oplus \mathbf{X} \oplus \mathbf{G}$ where $\mathbf{x}$ is the leading free variable. By definition of unsafeness, $\mathbf{x}_\ell >_H \mathbf{G}$ and so the adversary is able to learn $[\![\sigma G]\!]$ from the information it has received from the encryptor before it computes $[\![\sigma x_\ell]\!]$. Thus, we can let $\sigma x_\ell = \sigma G$ and $\sigma x' = 0$ for all summands $\mathbf{x'}$ of $\mathbf{X}$, to obtain $[\![\sigma x_\ell \oplus X \oplus G]\!] = 0$ with probability 1. ☐

We now lay the groundwork for an indistinguishability result for safe cryptosystems. The following lemma allows us to put a bound on the probability of *bad* occuring in either G_{str} or G_{symb}.

Lemma VI.1. *Suppose that a* $MOO_\oplus$ *cryptosystem* $\mathbb{C}$ *is safe. The probability, for a given symbolic history* $\mathbf{H}$, *that bad is set to 1 for security parameter* λ *in either* G_{str} *or* G_{symb} *is bounded by* $q_{\mathbf{H}}(q_{\mathbf{H}} - 1)2^{-\lambda-1}$, *where* $q_{\mathbf{H}}$ *is the number of function calls to* f_{str} *(respectively to* f_{symb}*) made by the encryptor while executing* $\mathbf{H}$.

Proof. We compute the bound for G_{symb}; it will be the same for G_{str}. The probability that *bad* is set to one is the probability that there are two terms $\mathbf{f}(\mathbf{u_1})$ and $\mathbf{f}(\mathbf{u_2})$ such that $[\![\sigma u_1 \oplus u_2]\!] = 0$. We break the proof down into two cases, depending on the $\mathbf{H}$-canonical form of $\mathbf{v}$.

1) The $\mathbf{H}$-canonical form of $\mathbf{u_1} \oplus \mathbf{u_2}$ is $\mathbf{G}$ where $\mathbf{G} \neq 0$

2) The $\mathbf{H}$-canonical form of $\mathbf{u_1} \oplus \mathbf{u_2}$ is $\mathbf{x}_\ell \oplus \mathbf{X} \oplus \mathbf{G}$, where $\mathbf{G} \neq 0$ and $\mathbf{G} \not\leq_{\mathbf{H}} \mathbf{x}_\ell$

To compute the probability of a collision in the first case, we use the fact that, by the construction of the G_{symb} encryptor, $[\![\sigma G]\!]$ is the exclusive-or of independently uniformly randomly generated strings, and thus is itself independently uniformly randomly generated. Thus, the probability that $[\![G]\!] = 0$ is $2^{-\lambda}$. For the second case, we use in addition the fact that the adversary has not received the output of σG or any function of the output by the time it computes σx_ℓ. Thus, the probability that $[\![\sigma x_\ell]\!] = [\![\sigma(X \oplus G)]\!]$ is again $2^{-\lambda}$. Summing over all the possible pairs of $\mathbf{f}$-rooted terms, we get a bound of $q_{\mathbf{H}}(q_{\mathbf{H}} - 1)2^{-\lambda-1}$. □

We are now almost in a position to prove a result about IND\$-CPA security. But first we need the following definition, which allows us to rule out trivially insecure systems.

Definition VI.5. We say that a symbolic history $\mathbf{H}$ is *degenerate* if there is a subset $\mathbf{D}$ of terms exchanged in $\mathbf{H}$ such $\sum_{\mathbf{d} \in \mathbf{D}} \oplus \mathbf{d} =_\oplus 0$. We say that a $MOO_\oplus$ cryptosystem $\mathbb{C}$ is *degenerate* if there is a degenerate symbolic history $\mathbf{H}$ obtainable from $\mathbb{C}$.

We note that even obviously insecure modes can be nondegenerate. Consider for example Electronic Codebook (ECB) mode, in which every symbolic history is of the form $[\mathbf{x_1}.\mathbf{f}(\mathbf{x_1}).\mathbf{x_2}.\mathbf{f}(\mathbf{x_2}).\ldots.\mathbf{x_k}.\mathbf{f}(\mathbf{x_k})]$. It is clear that there is no subset $\mathbf{D}$ of the terms exchanged in a symbolic history such that $(\sum_{\mathbf{x_i} \in \mathbf{D}} \oplus \mathbf{x_i}) \oplus (\sum_{\mathbf{f}(\mathbf{x_j}) \in \mathbf{D}} \oplus \mathbf{f}(\mathbf{x_j})) =_\oplus 0$. On the other hand, any mode that can produce a symbolic history of the form $\nu\mathbf{r}[\mathbf{x_1}.\mathbf{x_1} \oplus \mathbf{r}.\mathbf{x_2}.\mathbf{x_2} \oplus \mathbf{r}]$ is degenerate, since $\mathbf{x_1} \oplus (\mathbf{x_1} \oplus \mathbf{r}) \oplus \mathbf{x_2} \oplus (\mathbf{x_2} \oplus \mathbf{r}) =_\oplus \mathbf{0}$.

Lemma VI.2. *No degenerate $MOO_\oplus$ cryptosystem is* IND\$-CPA *secure.*

Proof. This follows easily from the fact that, if $\sum_{\mathbf{d} \in \mathbf{D}} \oplus \mathbf{d} =_\oplus 0$, then $\sum_{\mathbf{d} \in \mathbf{D}} \oplus [\![\ \sigma d]\!]$ with probability 1 for any substitution σ. □

We now show that any safe, non-degenerate $MOO_\oplus$ cryptosystem is IND\$-CPA secure. First, we show that, for any non-degenerate cryptosystem, the output of the f_{symb} encryptor is a random string of bits. That is, in this case, the G_{symb} encryptor also serves as a random encryptor.

Lemma VI.3. *Let $\mathbf{H}$ be a non-degenerate symbolic history for a $MOO_\oplus$ program $\mathbb{C}$. Then, for any substitution σ, the output of the G_{symb} encryptor in $[\![\sigma H]\!]$ is a uniformly distributed string. In particular, if $\mathbb{C}$ is a nondegenerate cryptosystem, every output of the G_{symb} encryptor is a uniformly distributed string. In other words, the adversary has zero advantage in distinguishing between the G_{symb} encryptor and the random encryptor.*

Proof. Let the terms returned by the encryptor in $\mathbf{H}$ be $\{\mathbf{X_1} \oplus\ \mathbf{G_1}; \cdots; \mathbf{X_k} \oplus \mathbf{G_k}\}$. Let σ be a substitution computed by the adversary, and let $\{[\![\sigma X_1 \oplus \sigma G_1]\!], \cdots, [\![\sigma X_k \oplus \sigma G_k]\!]\}$ be the output of the encryptor in Game G_{symb}. By the hypothesis there is no $\oplus$ equation satisfied by the terms $\mathbf{G_i}$, so the blocks $[\![\sigma G_i]\!]$ are independently uniformly randomly distributed. Moreover, also by the hypothesis, the adversary has no advance knowledge of G_i by the time it needs to compute σ on the leading variable of $X_i \oplus G_i$. Thus the bitstirngs $[\![\sigma X_i \oplus \sigma G_i]\!]$ are independently uniformly distributed as well, and so the sequence of bitstrings is a uniformly distributed random string. □

Theorem VI.1. *Any safe, non-degenerate $MOO_\oplus$ cryptosystem is* IND\$-CPA *secure. In particular, the adversary's advantage in distinguishing the between the real and the random encryptor is bounded by $q(q-1)2^{-\lambda-1}$, where q is the maximum number of calls to f made by the encryptor.*

Proof. The bound follows from the fact that by Lemma VI.1 the advantage of the adversary in distinguishing between the G_{str} encryptor and the G_{symb} encryptor is bounded by $q(q-1)2^{-\lambda-1}$ and by Lemma VI.3 its advantage in distinguishing between the G_{symb} encryptor and the random encryptor is 0. IND\$-CPA security then follows from the fact that the maximum number of calls made by the encryptor the f is bounded by a polynomial function of λ. □

We illustrate these results with an example.

Example VI.3. Consider messagewise CBC encryption, in which ciphertext is not sent to the adversary until all the plaintext blocks have been received. Let $\mathbf{H}$ be a symbolic history. Since all the terms in any symbolic history of CBC are different, CBC is nondegenerate. We now let $\mathbf{f}(\mathbf{u_1})$ and $\mathbf{f}(\mathbf{u_2})$ be two different f-rooted terms appearing in $\mathbf{H}$. Without loss of generality we may assume that $\mathbf{f}(\mathbf{u_1})$ is sent to the adversary before $\mathbf{f}(\mathbf{u_2})$. Then by the definition of CBC encryption we have $\mathbf{u_i} = \mathbf{x_i} \oplus \mathbf{g_i}$, where $\mathbf{g_i}$ is either $\mathbf{r_i}$ or $\mathbf{f}(\mathbf{v_i})$. Since $\mathbf{g_i}$ is not sent to the adversary until after it sends $\mathbf{x_i}$, we have no variable summand of the $\mathbf{H}$-canonical expression $\mathbf{x_1} \oplus \mathbf{x_2} \oplus \mathbf{g_1} \oplus \mathbf{g_2}$ such that $\mathbf{x_i} >_{\mathbf{H}} \mathbf{g_1} \oplus \mathbf{g_2}$. It follows that $\mathbf{H}$ is safe. Thus CBC encryption is IND\$-CPA-secure.

VII. Necessary and Sufficient Conditions for IND\$-CPA Security

In the previous section we showed that safety is a sufficient condition for IND\$-CPA security of a $MOO_\oplus$ cryptosystem. However, it is it is not necessary. In this section we show how safety can be used both to define a symbolic condition on sequences of unsafe pairs that is necessary and sufficient for IND\$-CPA, and to prove its necessity and sufficiency.

First, we show that it may be possible for unsafe pairs to exist in a cryptosystem and for the output to still be indistinguishable from random.

Example VII.1. Let $\mathbb{C}$ be a $MOO_{\oplus}$ cryptosystem that encrypts four-block long messages $\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}$, according to the symbolic history $\mathbf{H}_{\mathbb{C}}$ below:

$$\begin{aligned}\mathbf{H}_{\mathbb{C}} =& \mathbf{r_3}.\mathbf{x_1}.\mathbf{x_2}.\mathbf{f}(\mathbf{x_1} \oplus \mathbf{r_1}) \oplus \mathbf{f}(\mathbf{x_2} \oplus \mathbf{r_2}).\mathbf{r_1}.\mathbf{x_3}.\mathbf{x_4}.\\ &\mathbf{f}(\mathbf{x_3} \oplus \mathbf{r_3}) \oplus \mathbf{f}(\mathbf{x_4} \oplus \mathbf{r_4})\mathbf{r_2}.\mathbf{r_4}\end{aligned}$$

$\mathbf{H}_{\mathbb{C}}$ contains an unsafe pair $(\mathbf{f}(\mathbf{x_1} \oplus \mathbf{r_1}), \mathbf{f}(\mathbf{x_3} \oplus \mathbf{r_3}))$ with a computable unifier $\sigma\mathbf{x_3} = \mathbf{x_1} \oplus \mathbf{r_1} \oplus \mathbf{r_3}$, giving us

$$\begin{aligned}\sigma\mathbf{H}_{\mathbb{C}} =_{\oplus}& \mathbf{r_3}.\mathbf{x_1}.\mathbf{x_2}.\mathbf{f}(\mathbf{x_1} \oplus \mathbf{r_1}) \oplus \mathbf{f}(\mathbf{x_2} \oplus \mathbf{r_2}).\mathbf{r_1}.\mathbf{x_4}\\ &\mathbf{f}(\mathbf{x_1} \oplus \mathbf{r_1}) \oplus \mathbf{f}(\mathbf{x_4} \oplus \mathbf{r_4}).\mathbf{r_2}.\mathbf{r_4}\end{aligned}$$

(leaving out $\sigma\mathbf{x_3}$, which is now redundant). We note that the encryptor's output in $\sigma H_{\mathbb{C}}$ is indistinguishable from random, because it is nondegenerate and because every pair of f-rooted terms is safe. Indeed, every encrypted block is "protected" by an f-rooted term that does not belong to an unsafe pair, even after other unsafe pairs are unified. Suppose moreover that multiple copies of $\sigma H_{\mathbb{C}}$ with fresh variables are interleaved obtain a symbolic history $\mathbf{H}'$. Consider any pair of terms $(\mathbf{y} \oplus \mathbf{a})$ and $(\mathbf{z} \oplus \mathbf{b})$ and assume, without loss of generality, that $(\mathbf{z} \oplus \mathbf{b})$ is sent either in the same term as $(\mathbf{y} \oplus \mathbf{a})$ or in a later one. Then $\mathbf{b}$ will not be sent to the adversary until after it has sent both $\mathbf{y}$ and $\mathbf{z}$ to the encryptor. Thus the pair $\{(\mathbf{y} \oplus \mathbf{a}), (\mathbf{z} \oplus \mathbf{b})\}$ is safe.

On the other hand, it is also possible for an unsafe symbolic history to fail to be IND\$-CPA secure, even though unification of a single unsafe pair does not result in a degenerate history. That it is, we might require first an unsafe pair, then a substitution unifying the members of the unsafe pair that produces another unsafe pair, and so on.

Example VII.2.

$$\begin{aligned}\mathbf{H} =& \mathbf{x_1}.\mathbf{r_1}.\mathbf{r_2}.\mathbf{x_2}.\mathbf{f}(\mathbf{f}(\mathbf{x_1} \oplus \mathbf{r_1}) \oplus \mathbf{f}(\mathbf{x_2} \oplus \mathbf{r_2})).\\ &\mathbf{r_3}.\mathbf{x_3}.\mathbf{f}(\mathbf{x_3} \oplus \mathbf{r_3})\end{aligned}$$

Note that $\{\mathbf{f}(\mathbf{f}(\mathbf{x_1} \oplus \mathbf{r_1}) \oplus \mathbf{f}(\mathbf{x_2} \oplus \mathbf{r_2})), \mathbf{f}(\mathbf{x_3} \oplus \mathbf{r_3})\}$ is a safe pair. However, $\{\mathbf{f}(\mathbf{x_1} \oplus \mathbf{r_1}), \mathbf{f}(\mathbf{x_2} \oplus \mathbf{r_2})\}$ is an unsafe pair, with $\mathbf{H}$-unifier $\sigma\mathbf{x_2} = \mathbf{x_1} \oplus \mathbf{r_1} \oplus \mathbf{r_2}$. Making the substitution, and removing the now redundant variable $\mathbf{x_2}$ sent by the adversary, we have the sequence:

$$\sigma\mathbf{H} =_{\oplus} \mathbf{x_1}.\mathbf{r_1}.\mathbf{r_2}.\mathbf{f}(\mathbf{f}\mathbf{x_1} \oplus \mathbf{r_1}) \oplus \mathbf{f}(\mathbf{0}).\mathbf{r_3}.\mathbf{x_3}.\mathbf{f}(\mathbf{x_3} \oplus \mathbf{r_3})$$

where $\{\mathbf{f}(\mathbf{f}(\sigma\mathbf{x_1} \oplus \mathbf{r_1}) \oplus \mathbf{f}(\sigma\mathbf{x_2} \oplus \mathbf{r_2})), \mathbf{f}(\sigma\mathbf{x_3}) \oplus \mathbf{r_3})\} = \{\mathbf{f}(\mathbf{0}), \mathbf{f}(\mathbf{x_3} \oplus \mathbf{r_3})\}$ is an unsafe pair in $\sigma\mathbf{H}$, with $\sigma\mathbf{H}$-unifier $\pi\mathbf{x_3} = \mathbf{r_3}$. The resulting symbolic history is degenerate:

$$\pi\sigma\mathbf{H} =_{\oplus} \mathbf{x_1}.\mathbf{r_1}.\mathbf{r_2}.\mathbf{f}(\mathbf{0}).\mathbf{r_3}.\mathbf{x_3}.\mathbf{f}(\mathbf{0})$$

In the rest of this section, we describe a symbolic condition on sequences of unsafe pairs that is necessary and sufficient for IND\$-CPA, and we prove its necessity and sufficiency. We first get some necessary definitions out of the way.

Definition VII.1. Let $\mathbf{H}$ be a symbolic history, and let π be a symbolic substitution on the variables of $\mathbf{H}$. We define $\mathbf{\Gamma}_{(\mathbf{H},\pi)}$ to be the sequence of terms constructed from $\pi\mathbf{H}$ in the following way:

1) If $\mathbf{x}$ is sent by the adversary in $\mathbf{H}$ such that π is not the identity on $\mathbf{x}$, remove $\pi\mathbf{x}$ from the list of plaintext blocks sent by the adversary.
2) Reduce all terms in $\pi\mathbf{H}$ to $\oplus$ normal form.

Let n be the number of free variables in $\mathbf{\Gamma}_{(\mathbf{H},\pi)}$. We define $\mathbb{C}_{(\mathbf{H},\pi)}$ to be the cryptosystem that only encrypts n-block messages, such that the symbolic history of an encryption of an single n-block message is $\mathbf{\Gamma}_{(\mathbf{H},\pi)}$). We define $\mathbf{unsafe}(\pi, \mathbf{H})$ to be the set of pairs $\{\mathbf{f}(\mathbf{u}), \mathbf{f}(\mathbf{v})\}$ such that $\mathbf{f}(\mathbf{u})$ and $\mathbf{f}(\mathbf{v})$ are elements of $Sub(\mathbf{H})$, $\{\pi\mathbf{f}(\mathbf{u}), \pi\mathbf{f}(\mathbf{v})\}$ is unsafe with respect to $\mathbf{\Gamma}_{(\mathbf{H},\pi)}$, and $\pi\mathbf{u} \neq_{\oplus} \pi\mathbf{v}$.

We are now ready to define the idea of a *solvable sequence*, which is a key tool used in the proof of the main theorem of this paper.

Definition VII.2. Let $\mathbb{C}$ be a $MOO_{\oplus}$ cryptosystem, and let $\mathbf{H}$ be a symbolic history associated with $\mathbb{C}$. Let $\mathbf{L}$ be a sequence of pairs $\{\mathbf{f}(\mathbf{u}), \mathbf{f}(\mathbf{v})\}$ such that $\mathbf{f}(\mathbf{u})$ and $\mathbf{f}(\mathbf{v})$ are elements of $Sub(\mathbf{H})$. We say that $\mathbf{L}$ is a *solvable* $\mathbf{H}$*-sequence* and that the substitution π is its *solution* if $\mathbf{L}$ is either empty, in which case π is the identity, or it is constructed the following way:

Suppose that $(\mathbf{L}$ is a solvable $\mathbf{H}$-sequence. If $\mathbf{unsafe}_{\pi,\mathbf{H}} \neq \emptyset$, pick an element $\{\mathbf{f}(\mathbf{u}), \mathbf{f}(\mathbf{v})\}$ from it, and let $\mathbf{x}_{\ell} \oplus \mathbf{X} \oplus \mathbf{G}$ be the $\mathbf{\Gamma}_{(\mathbf{H},\pi)}$-canonical form of $\{\pi u \oplus \pi v\}$. By the definition of unsafe we have $\mathbf{x}_{\ell} >_{\mathbf{\Gamma}_{(\mathbf{H},\pi)}} \mathbf{G}$. Let $\tau\mathbf{x}_{\ell} = \mathbf{X} \oplus \mathbf{G}$. Then $(\mathbf{L}.\{\mathbf{f}(\mathbf{u}), \mathbf{f}(\mathbf{v})\})$ is also a solvable $\mathbf{H}$-sequence, with solution $\tau\pi$.

We say that $(\mathbf{L})$ with solution π is a *maximal solvable* $\mathbf{H}$*-sequence* if $\mathbf{unsafe}_{\pi,\mathbf{H}} = \emptyset$.

We note that by construction any solvable sequence $\mathbf{L}$ has one and only one solution, which we can thus denote as $\pi_{\mathbf{L}}$. We also note however, that some solvable sequences can give the same solution. Consider the symbolic sequence $\mathbf{r}.\mathbf{x_1}.\mathbf{f}(\mathbf{x_1} \oplus \mathbf{r}).\mathbf{x_2}.\mathbf{f}(\mathbf{x_2} \oplus \mathbf{r}).\mathbf{f}(\mathbf{x_1} \oplus \mathbf{x_2})$. We note that both $L_1 = \{\mathbf{f}(\mathbf{x_1} \oplus \mathbf{r}), \mathbf{f}(\mathbf{x_2} \oplus \mathbf{r})\}$ and $L_2 = \{\mathbf{f}(\mathbf{x_1} \oplus \mathbf{r}), \mathbf{f}(\mathbf{x_1} \oplus \mathbf{x_2})\}$ are solvable sequences, and they both have they same solution, $\pi_{\mathbf{L}}\mathbf{x_2} = \mathbf{x_1}$.

Definition VII.3. Let $\mathbf{H}$ be a symbolic sequence. We define **solve(H)** to be the set of substitutions π such that $\pi = \pi_{\mathbf{L}}$ for some solvable $\mathbf{H}$ sequence $\mathbf{L}$.

In the following example, we show that the first equation in a solvable $\mathbf{H}$ sequence is not necessarily the one with the earliest leading variable.

Example VII.3. Consider a symbolic history $\mathbf{H} = [\mathbf{r_0}.\mathbf{f}(\mathbf{0}).\mathbf{x_0}.\mathbf{x_1}.\mathbf{f}(\mathbf{x_0} \oplus \mathbf{f}(\mathbf{x_1} \oplus \mathbf{r_0}))]$. Let $\mathbf{L}$ be the sequence $\{[\{\mathbf{x_1} \oplus \mathbf{r_0}, \mathbf{0}\}.\{\mathbf{x_0} \oplus \mathbf{f}(\mathbf{x_1} \oplus \mathbf{r_0}), \mathbf{0}\}]$. Then $\{\mathbf{f}(\mathbf{0}), \mathbf{f}(\mathbf{x_1} \oplus \mathbf{r_0})\}$ is unsafe, and if we let $\pi\mathbf{x_1} = \mathbf{r_0}$, then

$$\begin{aligned}&[\mathbf{r_0}.\mathbf{f}(\mathbf{0}).\mathbf{x_0}.\mathbf{x_1}.\mathbf{f}(\mathbf{x_0} \oplus \mathbf{f}(\pi\mathbf{x_1} \oplus \mathbf{r_0})] = \\ &[\mathbf{r_0}.\mathbf{f}(\mathbf{0}).\mathbf{x_0}.\mathbf{x_1}.\mathbf{f}(\mathbf{x_0} \oplus \mathbf{f}(\mathbf{x_0} \oplus \mathbf{r_0}))]\end{aligned} \quad (1)$$

which gives an unsafe pair $\{\mathbf{f}(\mathbf{0}), \mathbf{f}(\mathbf{x_0} \oplus \mathbf{f}(\mathbf{0}))\}$with unifier $\pi\mathbf{x_0} = \mathbf{f}(\mathbf{0})$. An adversary would implement this by first sending $[\![\pi x_0]\!] = [\![f(0)]\!]$ to the encryptor, anticipating that it will later send $[\![\pi x_1]\!] = [\![r_0]\!]$.

Adversarial strategies, as we shall see, can be closely related to solvable sequences and their solutions , because they provide the only place in which the adversary has a choice as to whether or not to unify two $\mathbf{f}$-rooted terms. Given a solvable sequence $\mathbf{L}$, with solution $\pi_{\mathbf{L}}$ the adversary can choose to unify all pairs in that sequence, but not to unify any unsafe pairs in $\pi\mathbf{H}$ that are not unified by $\pi_{\mathbf{L}}$. As it turns out, it is possible to consider any adversary as a collection of simpler adversaries who follow exactly this policy, each one for for a different member of $\mathbf{solv}(\mathbf{H})$. The following definition and lemma will allow us to do that.

We now want to determine the conditions on solvable sequences that guarantee IND\$-CPA security. We start by breaking down the substitution computed by the adversary into simpler adversaries that we can reason about more easily.

Definition VII.4. Let $\mathbf{H}$ be a symbolic history with computational realization H. If σ is a substitution to the free variables in H and $\pi \in \mathbf{solv}(\mathbf{H})$ we define the substitution σ_π as follows:

1) First compute $[\![\sigma H]\!]$.
2) Output $[\![\sigma H]\!]$ if and only if
   a) $[\![\sigma f(u)]\!] = [\![\sigma f(v)]\!]$ for all pairs $\{\mathbf{f}(\mathbf{u}), \mathbf{f}(\mathbf{v})\}$ such that $\sigma\mathbf{f}(\mathbf{u}) = \sigma\mathbf{f}(\mathbf{v})\}$ and;
   b) $[\![\sigma f(u)]\!] \neq [\![\sigma f(v)]\!]$ if $\pi\mathbf{f}(\mathbf{u}) \neq \pi\mathbf{f}(\mathbf{v})$ but $\{(\sigma\mathbf{f}(\mathbf{u}), \sigma\mathbf{f}(\mathbf{v}))\}$ is an unsafe pair with respect to $\pi\mathbf{H}$
3) Otherwise output $\perp$.

Lemma VII.1. *Let* $\mathbf{H}$ *be a symbolic history, let* σ *be a substitution to the variables of* $\mathbf{H}$*, and let* n *be the number of blocks sent in* $\mathbf{H}$*. For any set* C *of bitstrings of length* $\lambda \cdot n$*, we have*

$$\begin{aligned} &Pr([\![\sigma H]\!] \in C) = \\ &\quad \sum_{\pi \in \mathbf{solv}(\mathbf{H})} (Pr([\![\sigma_\pi \neq \perp) \wedge ([\![\sigma_\pi]\!] H]\!] \in C)) = \\ &\quad \sum_{\pi \in \mathbf{solv}(\mathbf{H})} Pr([\![\sigma_\pi H]\!] \neq \perp) \cdot \\ &\qquad\qquad Pr(([\![\sigma_\pi H]\!] \in C) \mid ([\![\sigma_\pi H]\!] \neq \perp)) \end{aligned}$$

Moreover, for any $\pi \in \mathbf{solv}(\mathbf{H})$*, we have* $[\![\sigma_\pi H]\!] = [\![\sigma_\pi \pi H]\!]$ *whenever* $[\![sigma_\pi]\!] \neq \perp$.

Proof. The equalities follow straightforwardly from the observation that any output $[\![\sigma H]\!]$ could have been output by a substitution σ_π for one and only one element π of $\mathbf{solv}(\mathbf{H})$. The last statement follows from the fact that when $[\![\sigma_\pi]\!] \neq \perp$ it satisfies all the equations induced by π, so nothing is changed by solving these equations first and then applying σ_π to the remaining variables. □

Theorem VII.1. *Let* $\mathbb{C}$ *be a* $MOO_\oplus$ *cryptosystem, such that for every symbolic history* $\mathbf{H}$*, and every* $\pi \in \mathbf{solv}(\mathbf{H})$*,* $\mathbf{\Gamma}_{(\mathbf{H},\pi)}$ *is nondegenerate. Then the adversary's advantage in the* IND\$-CPA*-game is negligible. In particular, it is bounded by* $q(q-1)2^{-\lambda-1}$*, where* q *is the maximum number of calls the encryptor may make to the* f *function.*

Proof. Let $\mathbf{H}$ be a symbolic history, and let n be its length, and let σ be the substitution computed the adversary. Let C be the set of possible outputs of σH exhibiting a collision. If we can show that $PR(([\![\sigma_\pi H]\!] \in C) \mid ([\![\sigma_\pi H]\!] \neq \perp)) \leq q(q-1)2^{-\lambda-1}$ for any $\pi \in solv(H)$, we than have, by Lemma VII.1:

$$\begin{aligned} &Pr([\![\sigma H]\!] \in C) \leq \\ \sum_{\pi \in \mathbf{solv}(\mathbf{H})} &Pr([\![\sigma_\pi H]\!] \neq \perp) \cdot q(q-1)2^{-\lambda-1} = \\ (\sum_{\pi \in \mathbf{solv}(\mathbf{H})} &Pr([\![\sigma_\pi H]\!] \neq \perp)) \cdot q(q-1)2^{-\lambda-1} = \\ &1 \cdot q(q-1)2^{-\lambda-1} = \\ &q(q-1)2^{-\lambda-1} \end{aligned}$$

Thus, it is enough to show that, for any $\mathbf{H}$, any $\pi \in \mathbf{solv}(\mathbf{H})$, and any substitution σ to the free variables of H, it is enough to show that $Pr(([\![\sigma_\pi H]\!] \in C) \leq q(q-1)2^{-\lambda-1}$whenever $[\![\sigma_\pi H]\!] \neq \perp$. We also note that, again by Lemma VII.1 we have $[\![\sigma_\pi]\!] = [\![\sigma_\pi \pi]\!]$ wherever $\sigma_\pi \neq \perp$, so it is enough to prove the bound when we replace $\mathbf{H}$ with $\mathbf{\Gamma}_\pi$ and work with the cryptosystem $\mathbb{C}_{(\mathbf{H},\pi)}$.

As before, we will compare the games G_{str} and G_{symb}. Since by assumption $\mathbf{\Gamma}_{(\mathbf{H},\pi_{\mathbf{L}})}$ is nondegenerate, the output of $\sigma_{(H,\pi_L)}\Gamma_{(H,\pi_L)}$ is a randomly generated string by Lemma VI.3. Thus, all we have to do to show that the bound holds is to show that it also bounds the probability of *bad* occurring in G_{symb} when the substitution $\sigma_{(H,\pi_L)}$ is used.

We note that by definition $\pi_{\mathbf{L}}\mathbf{u} = \pi_{\mathbf{L}}\mathbf{v}$ when $(\mathbf{f}(\mathbf{u}), \mathbf{f}(\mathbf{v})) \in \mathbf{L}$, and, if $(\mathbf{f}(\mathbf{u}), \mathbf{f}(\mathbf{v}) \in \mathbf{unsafe}(\mathbf{\Gamma}_{(\mathbf{H},\pi_{\mathbf{L}})})$ then $[\![\sigma_{(\mathbf{H},\mathbf{L},\pi)}u]\!] \neq [\![\sigma_{(\mathbf{H},\mathbf{L},\pi)}v]\!]$ with probability 1. Thus, *bad* can happen if and only if $[\![\sigma_{(H,\pi_L)}u]\!] = [\![\sigma \mid (H,\pi_L)v]\!]$ where $(\mathbf{f}(\mathbf{u}), \mathbf{f}(\mathbf{v}))$ is safe with respect to $\mathbf{\Gamma}_{(\mathbf{H},\pi_{\mathbf{L}})}$. But in that case the probability that $[\![\sigma_{(H,\pi_L)}u]\!] = [\![\sigma_{(H,\pi_L)}v]\!]$ is $2^{-\lambda}$. Thus the probability of *bad* occurring in the construction of $[\![\sigma_{(\mathbf{H},\mathbf{L},\pi_{\mathbf{L}})}H]\!]$ is bounded above by $q(q-1)2^{-\lambda-1}$ where q is the maximal number of f-rooted terms computable by the encryptor. It follows from Lemma VII.1 that the adversary's advantage in distinguishing G_{str} from G_{symb} is also bounded by $q(q-1)2^{-\lambda-1}$. The conclusion that the adversary's advantage is negligible follows from the assumption that the encryptor is polynomial time, so the maximal number of calls to the f function must be bounded by a polynomial function of λ.

□

VIII. Extending Results to Modes Using Additional Functions

We have proven results for modes making use of a minimal number of operations, but of course many modes use more than that. Here we discuss some other operations and properties of modes, and how they might be handled using the approach developed in this paper.

a) Hash Functions: These would be modeled as random functions, although functions computable by the adversary as well as the encryptor.

b) Increment by One: This is a function that is used, for example in Counter Mode to provide variability of cipher blocks. If s is a λ-bit bitstring, then the increment-by-one function computes $s + 1$ modulo λ. This inc function can generally be modeled as a free unary function symbol. However, there are some pitfalls when other field or group operations are used. For example, consider the case when $\oplus$ and inc are used in the same cryptosystem. Then, for a randomly chosen string s, $inc(s) = s \oplus 1$ with probability $1/2$, leading to potential attacks. This can be avoided by putting the appropriate constraints on the use of $\oplus$ and inc in the same term, as is done by Malozemoff et al. in [12]. Since our conditions on $MOE_{\oplus}$ unification when only $\oplus$ is used already lead to constraints, we can potentially deal with these new constraints the same way we do the original ones. However, in the case that a constraint is violated it may be harder to tell whether or not security is also violated.

c) Finite Field and Group Operations: If only one of these operations is used, these should be fairly straightforward to deal with (although avoiding the divide-by-zero problem in the finite field case may be an issue). However, the same issues crop up as in Increment by One and $\oplus$ if more than one is used: two terms that are not equal in the term algebra may have computational interpretations that are not equal with non-negligible probability. Thus, similar constraints will be as in the Increment by One case will need to be used.

d) Concatenation: Concatenation has generally been difficult to reason about symbolically. For one thing, there are constraints on length that need to be met. The concatenation of two λ-length blocks is not another λ-length block. Moreover, concatenation is associative, a difficult property to reason about when the strings concatenated can be any length; one may end up with an infinite set of results. However, in modes (and other cryptosystems), the number of possible lengths of strings to be concatenated is fairly small, and it may be possible to treat terms with lengths as a kind of type system. This should make the problem more tractable.

e) Block Ciphers With Tweaks: Many block ciphers are not deterministic; they have *tweaks* [11] that are used to provide variability of ciphertext output, so that this is no longer the responsibility of the mode. A tweak is generally known to the adversary, however, and may even be predictable. The main criterion is that no two tweaks be the same. This is something we can model in our approach by introducing a new quantifier μ, such that no two variables quantified by μ may be the same, and two μ would be mapped to two different non-random bitstrings.

f) Decryption : In this paper, we have ignored decryption and concentrated on secrecy. However, it is necessary that any encryption algorithm should have the property that plaintext can be recovered from ciphertext by some algorithm that is able to compute the inverse of the block cipher. Since the probability of such an algorithm succeeding should be 1, this may be something that can be addressed completely at the symbolic level.

g) Authentication: Many cryptographic modes of operation provide authentication. Authentication provides security against forgery, so that an adversary should not be able to create a message that passes some kind of authentication test with non-negligible probability. In [8], Hoang et al. provide symbolic criteria for this property for a particular kind of authentication method used by a sizable class of authenticated cryptographic modes of operation. One question to ask here is, what other authentication methods could be reasoned about symbolically, and how?

IX. Conclusion and Open Problems

We have identified a simple, syntactically checkable condition that is sufficient to guarantee IND\$-CPA security for block cipher modes of operation. We have also identified a more complex condition that is both necessary and sufficient.

This work opens up a number of direction of future research. The first, of course, is to develop algorithms for checking these criteria, and for generating cryptosystems that satisfy these criteria. This work indeed has already begun. We have been working with colleagues on not only the above problems but on the complexity of the algorithmic problems that we have encountered. There are also many ways in which these results can be extended. One is to extend the range of our work on modes of operation, as was discussed in the last section. More generally, we can extend our research to similar problems, perhaps starting with the ones covered by the Linicrypt model [13], which provides a well-understood set of primitives to work with, and expanding our aim by adding additional primitives and security properties, as we discussed in Section VIII As in this work, our aim will be to identify and exploit problems that can be reduced to problems in analysis of term algebras, in particular problems in unification and disunification, that then can be addressed using symbolic techniques.

X. Acknowledgements

This work was funded by ONR Code 31.

References

[1] Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *Conference Record of POPL 2001: The 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, London, UK, January 17-19, 2001*, pages 104–115, 2001.

[2] Michael Backes and Birgit Pfitzmann. A cryptographically sound security proof of the needham-schroeder-lowe public-key protocol. In *FST TCS 2003: Foundations of Software Technology and Theoretical Computer Science, 23rd Conference, Mumbai, India, December 15-17, 2003, Proceedings*, pages 1–12, 2003.

[3] Gregory V. Bard. Blockwise-adaptive chosen-plaintext attack and online modes of encryption. In *Cryptography and Coding, 11th IMA International Conference, Cirencester, UK, December 18-20, 2007, Proceedings*, pages 129–151, 2007.

[4] Mathieu Baudet, Véronique Cortier, and Steve Kremer. Computationally sound implementations of equational theories against passive adversaries. In *Automata, Languages and Programming*, pages 652–663. Springer, 2005.

[5] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, pages 409–426, 2006.

[6] Brent Carmer and Mike Rosulek. Linicrypt: A model for practical cryptography. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, pages 416–445, 2016.

[7] Martin Gagné, Pascal Lafourcade, Yassine Lakhnech, and Reihaneh Safavi-Naini. Automated proofs of block cipher modes of operation. *J. Autom. Reasoning*, 56(1):49–94, 2016.

[8] Viet Tung Hoang, Jonathan Katz, and Alex J. Malozemoff. Automated analysis and synthesis of authenticated encryption schemes. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 84–95, 2015.

[9] Antoine Joux, Gwenaëlle Martinet, and Frédéric Valette. Blockwise-adaptive attackers: Revisiting the (in)security of some provably secure encryption models: Cbc, gem, IACBC. In *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, pages 17–30, 2002.

[10] Steve Kremer and Laurent Mazaré. Adaptive soundness of static equivalence. In *Computer Security–ESORICS 2007*, pages 610–625. Springer, 2007.

[11] Moses D. Liskov, Ronald L. Rivest, and David A. Wagner. Tweakable block ciphers. *J. Cryptology*, 24(3):588–613, 2011.

[12] Alex J Malozemoff, Jonathan Katz, and Matthew D Green. Automated analysis and synthesis of block-cipher modes of operation. In *Computer Security Foundations Symposium (CSF), 2014 IEEE 27th*, pages 140–152. IEEE, 2014.

[13] Ian McQuoid, Trevor Swope, and Mike Rosulek. Characterizing collision and second-preimage resistance in linicrypt. In *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part I*, pages 451–470, 2019.

[14] Catherine A. Meadows. Symbolic security criteria for blockwise adaptive secure modes of encryption. *IACR Cryptology ePrint Archive*, 2017:1152, 2017.

[15] Daniele Micciancio and Saurabh Panjwani. Adaptive security of symbolic encryption. In *Theory of Cryptography*, pages 169–187. Springer, 2005.

[16] Phillip Rogaway. Nonce-based symmetric encryption. In *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, pages 348–359, 2004.

[17] Dominique Unruh. The impossibility of computationally sound XOR. *IACR Cryptology ePrint Archive*, 2010:389, 2010.