An extended abstract of this paper appears in *Advances in Cryptology – Crypto 94 Proceedings*, Lecture Notes in Computer Science Vol. 839, Y. Desmedt ed., Springer-Verlag, 1994. This is the full version, to appear in *Journal of Computer and System Sciences*.

The Security of the Cipher Block Chaining Message Authentication Code

Mihir Bellare* Joe Kilian† Phillip Rogaway‡

August 7, 2000

Abstract

Let F be some block cipher (eg., DES) with block length l. The Cipher Block Chaining Message Authentication Code (CBC MAC) specifies that an m-block message $x = x_1 \cdots x_m$ be authenticated among parties who share a secret key a for the block cipher by tagging x with a prefix of y_m, where $y_0 = 0^l$ and $y_i = F_a(m_i \oplus y_{i-1})$ for $i = 1, 2, \ldots, m$. This method is a pervasively used international and U.S. standard. We provide its first formal justification, showing the following general lemma: cipher block chaining a pseudorandom function yields a pseudorandom function. Underlying our results is a technical lemma of independent interest, bounding the success probability of a computationally unbounded adversary in distinguishing between a random ml-bit to l-bit function and the CBC MAC of a random l-bit to l-bit function.

*Department of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. E-Mail: `mihir@cs.ucsd.edu`. URL: `http://www-cse.ucsd.edu/users/mihir`. Supported by NSF CAREER Award CCR-9624439 and a Packard Foundation Fellowship in Science and Engineering.

† NEC Research Institute, 4 Independence Way, Princeton, New Jersey 08540, USA. E-mail: `joe@research.nj.nec.com`.

‡ Department of Computer Science, University of California at Davis, Davis, CA 95616, USA. E-mail: `rogaway@cs.ucdavis.edu`. URL: `http://wwwcsif.cs.ucdavis.edu/~rogaway`. Supported by NSF CAREER Award CCR-9624560.

Contents

1 Introduction

1.1 The problem: Is the CBC MAC secure?

Message authentication lets communicating partners who share a secret key verify that a received message originates with the party who claims to have sent it. This is one of the most important and widely used cryptographic tools. It is most often achieved using a "message authentication code," or MAC. This is a short string $MAC_a(x)$ computed on the message x to be authenticated and the shared secret key a. The sender transmits $(x, MAC_a(x))$ and the receiver, who gets (x', σ'), verifies that $\sigma' = MAC_a(x')$.

The most common MAC is built using the idea of "cipher block chaining" some underlying block cipher. To discuss this approach we first need some notation. Given a function $f\colon \{0,1\}^l \to \{0,1\}^l$ and a number $m \geq 1$ we denote by $f^{(m)}\colon \{0,1\}^{ml} \to \{0,1\}^l$ the function that maps an ml-bit input $x = x_1 \cdots x_m$ (where $|x_i| = l$) to the l-bit string y_m computed as follows: set $y_0 = 0^l$ and then iterate $y_i \leftarrow f(y_{i-1} \oplus x_i)$ for $i = 1, \ldots, m$. We call $f^{(m)}$ the *(m-fold) cipher block chaining* of f. A block cipher F with key-length k and block-length l specifies a family of permutations $F_a\colon \{0,1\}^l \to \{0,1\}^l$, one for each k-bit key a. The CBC MAC constructed from F has an associated parameter $s \leq l$ which is the number of bits it outputs. The CBC MAC is then defined for any ml-bit string $x = x_1 \ldots x_m$ by

$$\mathsf{CBC}^m\text{-}F_a(x_1 \ldots x_m)[s] \stackrel{\text{def}}{=} \text{The first } s \text{ bits of } F_a^{(m)}(x_1 \cdots x_m)\ .$$

The CBC MAC is an international standard [13]. The most popular and widely used special case uses $F = \text{DES}$ (the Data Encryption Standard; here $k = 56$ and $l = 64$) and $s = 32$, in which case we recover the definition of the corresponding U.S. standard [2]. These standards are extensively employed in the banking sector and in other commercial sectors. Given this degree of usage and standardization, one might expect that there would be a large body of work aimed at learning if the CBC MAC is secure. Yet this has not been the case. In fact, prior to the current results it was seen as entirely possible that $\mathsf{CBC}^m\text{-}F$ might be a completely insecure MAC even when F is realized by a highly secure block cipher. There was no reason to believe that the internal structure of F could not "interact badly" with the specifics of cipher block chaining.

1.2 Our approach

In this paper we will show that CBC MAC construction is secure if the underlying block cipher is secure. To make this statement meaningful we need first to discuss what we mean by security in each case.

WHAT DOES IT MEAN TO ASSUME A BLOCK CIPHER IS SECURE? To describe the security of a block cipher we elaborate on a viewpoint suggested by Luby and Rackoff [15, 16] with regard to DES. They suggest that a good block cipher can be assumed to behave as a good pseudorandom function (PRF). The formal notion of a PRF is due to Goldreich, Goldwasser and Micali [10]. Roughly said, the security of a family of functions F as a PRF is measured by an adversary's inability to distinguish the following two types of objects, based on their input/output behavior: a black-box for $F_a(\cdot)$, on a random key a; and a black-box for a truly random function $f(\cdot)$.

A somewhat "tighter" model for a block cipher is to say that it should behave as a good pseudorandom permutation (PRP). The security of F as a PRP is measured by an adversary's inability to distinguish the following two types of objects: a black-box for $F_a(\cdot)$, on a random key a; and a black-box for a random permutation $\pi(\cdot)$.

What does it mean for a MAC to be secure? Our notion of security for a message authentication code adopts the viewpoint of Goldwasser, Micali and Rivest [12] with regard to signature schemes; namely, a secure MAC must resist existential forgery under an adaptive chosen-message attack. An adversary is allowed to obtain valid MACs of some number of messages of its choice, and wins if it can then output a "new" message (meaning one whose MAC it did not obtain during the chosen-message attack phase) together with a valid MAC of this message.

Concrete security. We wish to obtain results that are meaningful for practice. In particular, we aim to say something about the correct and incorrect use of block ciphers like DES. Such functions have finite domains; there is no asymptotics present. Thus we are led to avoid asymptotics and to specify security bounds quite concretely. We strive for reductions that are as security-preserving as possible, and we measure the degree of demonstrated security by way of explicit formulas—the "resource-translation functions."

We will only talk about finite families of functions and the resources needed to learn things about them. To any such family we associate an "insecurity function" that takes as input resource bounds for the adversary and returns a real number, this number being the maximum possible probability that an adversary could "break" the security of the family when restricted to the given resources. The meaning of "break" differs according to the security goal being considered: it might be in terms of pseudorandomness or as a MAC. The resources of interest are the running time t of the adversary and the number of queries q that she makes to an oracle that is her only point of access to $f(\cdot)$-values for the given instance f of family F. We emphasize the importance of keeping t and q separate: in practice, oracle queries (q) correspond to observations or interactions with a system whose overall structure often severely limits the reasonable values; but time (t) corresponds to off-line computation by the adversary, and cannot, therefore, be architecturally controlled. A proof of security for $\mathsf{CBC}^m\text{-}F$ is obtained by upper bounding the insecurity of $\mathsf{CBC}^m\text{-}F$ in terms of the insecurity of F itself. Let us look at some results to see how it works.

1.3 Results

Our main result is stated formally as Theorem 3.2. Informally, it says that the CBC-MAC transform is PRF-preserving. Namely, the CBC-MAC of a pseudorandom function (or permutation) F is itself a pseudorandom function. The security of the CBC-MAC as a MAC follows because it is a well-known observation that any PRF is a secure message authentication code [10, 11]—see Section 2.4 for details.

Statement. To illustrate the main result, let F be the given block cipher with block-length l. The concrete security statement of one version of our theorem is the following: for any integers $q, t, m \geq 1$,

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CBC}^m\text{-}F}(q,t) \quad \leq \quad \mathbf{Adv}^{\mathrm{prp}}_{F}(q',t') + \frac{q^2m^2}{2^{l-1}} \,, \qquad (1)$$

where $q' = mq$ and $t' = t + O(mql)$.

Explanation of terms. Here $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CBC}^m\text{-}F}(q,t)$ is the maximum, over all adversaries restricted to q input-output examples and execution time t, of the "advantage" that the adversary has (compared to simply guessing) in the game of distinguishing a random instance of family $\mathsf{CBC}^m\text{-}F$ from a random function of ml bits to l bits. Similarly, $\mathbf{Adv}_F^{\mathrm{prp}}(q,t)$ is the maximum, over all adversaries restricted to q' input-output examples and execution time t', of the "advantage" that the adversary has in the game of distinguishing a random instance of family F from a random permutation on l bits. Precise definitions of these quantities can be found in Section 2, but for the moment, it will suffice to remember that for these functions a small value corresponds to a lower breaking probability, and hence to greater security.

Qualitative interpretation. Roughly, Equation (1) says that the chance of breaking the CBC-MAC of F using some given amount of resources is not much more than the chance of breaking F itself with comparable resources. That is, if F is a secure PRP then $\mathsf{CBC}^m\text{-}F$ is a secure PRF. This qualitative statement already conveys information of a nature not found in previous approaches to the analysis of the CBC-MAC, because it demonstrates that if the underlying primitive is secure, than so is the MAC based on it. Thus there is no need to directly cryptanalyze the CBC-MAC; cryptanalytic effort can remain concentrated on the lower level primitive F. This is the benefit of the reductionist paradigm.

Quantitative interpretation. Practical information can be garnered by taking into account the quantitative aspects of the result. First note that no matter how secure the given block cipher F, our upper bound bound on the insecurity of $\mathsf{CBC}^m\text{-}F$ grows proportionally to the square of the number of queries times the square of the number of blocks in each message. If the security really drops off in such a manner, it is due to an inherent weakness in the CBC-MAC construction itself, and has nothing to do with the block cipher being used.

To assess the demonstrated security of the CBC-MAC using a given block cipher F we would use current cryptanalytic knowledge to estimate the value of $\epsilon' = \mathbf{Adv}_F^{\mathrm{prp}}(q',t')$ for given q',t'. Here ϵ' represents a probability of adversarial success that we can (for now) rule out. These values would of necessity be conjectural. With such a value for ϵ' in hand, we can compute the value of the right hand side of Equation (1) and thereby get a specific upper bound on the probability of adversarial success in breaking the CBC-MAC. Numerical examples will be given later in the paper.

Reductionist interpretation. The result can be interpreted in terms of adversary transformations as follows. Suppose there is an adversary A that breaks $\mathsf{CBC}^m\text{-}F$ with some probability ϵ while using resources q and t. Then A can be turned into an adversary A' of comparable time complexity t' that, making $q' = qm$ oracle queries, achieves advantage $\epsilon' = \epsilon - q^2m^2 \cdot 2^{-l+1}$ in breaking F itself.

Information theoretic case. The brunt of the proof addresses the information-theoretic case of the above result. Here we consider the problem of distinguishing a random ml-bit to l-bit function from the m-fold CBC of a random l-bit to l-bit function. In Theorem 3.1 we prove an absolute bound of $3q^2m^2 \cdot 2^{-l-1}$ on the advantage an adversary can derive. The bound holds irrespective of the adversary's running time.

Security as a mac. Theorem 4.1 completes the picture by upper bounding the MAC-insecurity of $\mathsf{CBC}^m\text{-}F$ in terms of the PRP-insecurity of F. This is done by combining the above result with Proposition 2.7, which shows that the standard PRF to MAC reduction has tight security. We also consider the tightness of our analysis.

1.4 Extensions and applications

Pseudorandom functions are basic tools in cryptography. In addition to shedding light on the security of the CBC MAC, our work provides a method of building secure PRFs that can be used in a wide range of applications, in the following way. Cryptographic practice directly provides PRFs (more accurately, PRPs) on fixed input lengths, in the form of block ciphers. On the other hand, PRFs are very useful in applications, but one typically needs PRFs on long strings. The CBC Theorem provides a provably-good way of extending the basic PRFs, which work on short inputs, to PRFs that work on longer inputs. It was based on such constructions that PRFs were suggested by [7] as the tool of choice for practical applications such as entity authentication and key distribution.

The CBC MAC of an l-bit block cipher provides an efficient way to produce a PRF to l-bits or fewer when the input is of fixed length ml. But often the input lengths may vary. We describe in Section 5 some simple mechanisms to extend the CBC MAC to authenticate words of arbitrary length. We also demonstrate that some plausible-looking mechanisms do not work, such as $MAC_a(x) = F_a^{(m+1)}(x \| m)$.

1.5 History and related work

The lack of any theorem linking the security of F to that of $F^{(m)}$ lead previous users of the CBC MAC to view $F^{(m)}$, and not F, as the basic primitive. For example when Bird et. al. [8] required a practical message authentication code in order to achieve their higher-level goal of entity authentication, they made appropriate assumptions about the CBC MAC.

A cryptanalytic approach directly attacks the CBC MAC based on details of the underlying block cipher F. An attempt to directly attack the DES CBC MAC using differential cryptanalysis is described in [17].

Another approach to studying MACs is rooted in the examination of protocols that use them. Stubblebine and Gligor [20] find flaws in the use of the CBC MAC in some well-known protocols. But as the authors make clear, the CBC MAC is not itself at fault for the indicated protocol failures; rather, it is the manner in which the containing protocols incorrectly embed the CBC MAC. The authors go on to correct some protocols by having them properly use the CBC MAC.

The concrete security approach makes more explicit and emphatic some features already present in the asymptotic approach typically used in theoretical works. With asymptotic analysis security guarantees often take the form "the success probability of a polynomially bounded adversary is negligible" (everything measured as a function of the security parameter). The concrete security can usually be derived by examining the proof. However, a lack of focus on getting good concrete security bounds has often led to reductions that are so inefficient that the results are of no obvious use to cryptographic practice.

1.6 Subsequent work

Since the appearance of the preliminary version of this work [4] there has been further related research.

The current paper provides an upper bound on the insecurity of the CBC MAC and our analysis highlights the role of collisions. Preneel and van Oorschot [19] give a corresponding attack, also exploiting collisions. Some gap remains between our result and theirs; closing it is an interesting problem. See Section 4 for more information. Another attack is given in [14].

Several CBC MAC variations have been suggested to get around the problem mentioned above that the CBC MAC is only secure when strings are of some one fixed length. One nice suggestion is to compute the (basic) CBC MAC using a first key, and then encipher that result using a second (independent) key. Petrank and Rackoff analyze this construction [18].

One might ask whether the security of $\mathsf{CBC}^m\text{-}F$ as a MAC could be shown to follow from a weaker assumption on F than that it is a PRF. Work of An and Bellare [1] shows that it is not enough to assume that F is a MAC; they give an example of a secure MAC F for which $\mathsf{CBC}^m\text{-}F$ is not a secure MAC.

Cipher block chaining is not the only method of constructing a MAC. Amongst the many proposed methods we mention XOR-MACs [6], HMAC [5] and UMAC [9]. Some of these alternative constructions improve on the CBC MAC in terms of speed or security bounds.

1.7 Discussion and open questions

Block ciphers like DES are in fact *permutations*. One open question is whether the permutativity of the block cipher could be exploited to prove a stronger reduction than that in our main theorem. The fact that one typically outputs a number of bits $s < l$ seems relevant and useful in strengthening the bounds that would otherwise be achieved.

This paper brings out the importance of modeling the fixed input and output lengths common to the primitives of contemporary cryptographic practice. When a family of functions, each from l bits to L bits (for some particular and fixed values of l and L) aims to mimic a family of random functions from l bits to L bits, we refer to this family as a *finite* PRF. Finite PRFs, and the concrete security analysis of constructions based on them, is a technique for investigating the efficacy of many classical (and not-so-classical) cryptographic constructions. In this way one can formally treat security constructs such as CBC encryption, finding for each such construction upper and lower bounds on its security [3].

2 Definitions and Basic Facts

The primitives we discuss in this paper include message authentication codes, pseudorandom functions and pseudorandom permutations. An important aspect of our approach is to use a "concrete" (sometimes also called "exact") security framework, meaning there are no asymptotics. This is necessary because we want to model block ciphers and their usages, and a block cipher is a finite object. In this section, we present definitions that enable a concrete security treatment. We also note basic facts or relations that we will exploit later.

2.1 Families of functions

All the above-mentioned objects are families of functions, having security properties differing from case to case. The starting point is thus to define (finite) families of functions. Security properties will be considered later.

A *family of functions* is a map $F\colon\ \mathit{Keys}(F) \times \mathit{Dom}(F) \to \mathit{Ran}(F)$. Here $\mathit{Keys}(F)$ is the key space of F; $\mathit{Dom}(F)$ is the domain of F; and $\mathit{Ran}(F)$ is the range of F. The two-input function

F takes a key $a \in \mathit{Keys}(F)$ and an input $x \in \mathit{Dom}(F)$ to return a point $F(a,x) \in \mathit{Ran}(F)$. If $\mathit{Keys}(F) = \{0,1\}^k$ for an integer k then we refer to k as the key-length. If $\mathit{Dom}(F) = \{0,1\}^d$ for some integer d then we refer to d as the input-length. If $\mathit{Ran}(F) = \{0,1\}^L$ for some integer L then we refer to L as the output-length. In this paper, $\mathit{Keys}(F)$, $\mathit{Dom}(F)$, and $\mathit{Ran}(F)$ will always be finite.

For each key $a \in \mathit{Keys}(F)$ we define the map F_a: $\mathit{Dom}(F) \to \mathit{Ran}(F)$ by $F_a(x) = F(a,x)$ for all $x \in \mathit{Dom}(F)$. Thus, F specifies a collection of maps from $\mathit{Dom}(F)$ to $\mathit{Ran}(F)$, each map being associated with a key. That is why F is called a family of functions. We refer to F_a as an *instance* of F.

We often speak of choosing a random key a uniformly from $\mathit{Keys}(F)$. This operation is written $a \stackrel{R}{\leftarrow} \mathit{Keys}(F)$. We write $f \stackrel{R}{\leftarrow} F$ for the operation $a \stackrel{R}{\leftarrow} \mathit{Keys}(F)$; $f \leftarrow F_a$. That is, $f \stackrel{R}{\leftarrow} F$ denotes the operation of selecting at random a function from the family F. When f is so selected it is called a *random instance* of F.

We say that F is a family of permutations if $\mathit{Dom}(F) = \mathit{Ran}(F)$, and for each key $a \in \mathit{Keys}(F)$ it is the case that F_a is a permutation (ie. a bijection) on $\mathit{Dom}(F)$.

Example. Any block cipher is a family of permutations. For example, DES is a family of permutations with $\mathit{Keys}(\mathrm{DES}) = \{0,1\}^{56}$ and $\mathit{Dom}(\mathrm{DES}) = \mathit{Ran}(\mathrm{DES}) = \{0,1\}^{64}$. This family has key-length 56, input-length 64, and output-length 64.

Random functions and permutations. In order to define PRFs and PRPs we first need to fix two function families. One is $\mathrm{Rand}^{l\to L}$, the family of all functions from $\{0,1\}^l$ to $\{0,1\}^L$, and the other is Perm^l, the family of all permutations on $\{0,1\}^l$. Before defining these objects formally, let us describe the intuition about their behavior that is important here. Consider an algorithm, A, that has an oracle for a random instance f of $\mathrm{Rand}^{l\to L}$ and makes some number of distinct queries to this oracle. Then every invocation of the oracle yields an output that is random and distributed independently of all previous outputs. If f is a random instance of Perm^l, then every invocation of the oracle yields an output distributed uniformly amongst all range points not already obtained as outputs of the oracle via previous queries.

Let us now specify these families formally. For this purpose it is convenient to fix some bijection ord_l: $\{0,1\}^l \to \{1,2,\ldots,2^l\}$, given for example by a canonical ordering of the elements of $\{0,1\}^l$. Now $\mathrm{Rand}^{l\to L}$: $\{0,1\}^k \times \{0,1\}^l \to \{0,1\}^L$ is a family with key-space $\{0,1\}^k$ where $k = L2^l$, and we interpret a key $a = a[1]\cdots a[2^l]$ in the key space as a sequence of L-bit strings that specifies the value of the associated function at each point in the input domain, meaning $\mathrm{Rand}^{l\to L}(a,x) = a[\mathrm{ord}_l(x)]$. The operation $f \stackrel{R}{\leftarrow} \mathrm{Rand}^{l\to L}$ simply selects a random function of l-bits to L-bits. On the other hand Perm^l: $\mathit{Keys}(\mathsf{Perm}^l) \times \{0,1\}^l \to \{0,1\}^l$ has a key space given by

$$\mathit{Keys}(\mathsf{Perm}^l) \quad = \quad \{\, (a[1],\ldots,a[2^l]) \ : \ a[1],\ldots,a[2^l] \in \{0,1\}^l \text{ are all distinct} \,\}\ ,$$

and for any key $a = (a[1],\ldots,a[2^l])$ in $\mathit{Keys}(\mathsf{Perm}^l)$ and any $x \in \{0,1\}^l$ we define $\mathsf{Perm}^l(a,x) = a[\mathrm{ord}_l(x)]$. The operation $f \stackrel{R}{\leftarrow} \mathsf{Perm}^l$ selects a random permutation on $\{0,1\}^l$.

2.2 Model of computation

We fix some particular Random Access Machine (RAM) as a model of computation. An adversary is a program for this RAM. An adversary may have access to an oracle. Its query-complexity, or the number of queries it makes, is the number of times it consults this oracle. When we speak of A's running time this will include A's actual execution time plus the length of A's description (meaning the length of the RAM program that describes A). This convention eliminates pathologies caused if one can embed arbitrarily large lookup tables in A's description.

(Alternatively, the reader can think in terms of circuits over some fixed basis of gates, like 2-input NAND gates. An adversary is such a circuit, and now time simply means the circuit size. Circuits are allowed special query gates for making oracle queries. This formulation is simpler to specify in full detail, but it is rather less intuitive.)

2.3 Pseudorandom functions and permutations

DISTINGUISHERS. The notion of a distinguisher is due to [10]. Let F^0: $\mathit{Keys}(F^0) \times D \to R$ and F^1: $\mathit{Keys}(F^1) \times D \to R$ be two function families with a common domain D and a common range R. A *distinguisher for* F^0 *versus* F^1 is an adversary A that has access to an oracle $f\colon\ D \to R$ and, at the end of its computation, outputs a bit. The oracle will be chosen either as a random instance of F^0 or as a random instance of F^1, and the distinguisher is trying to tell these cases apart. The "closer" the two families, the harder the task of the distinguisher, so that distinguishing ability provides a measure of "distance" between function families.

PRFS. The pseudorandomness of a function family F: $\mathit{Keys}(F) \times \{0,1\}^l \to \{0,1\}^L$ is its "distance" from the family of all functions. Namely, pseudorandomness measures the ability of a distinguisher to tell whether its given oracle is a random instance of F or a random function of $\{0,1\}^l$ to $\{0,1\}^L$.

Definition 2.1 Suppose F: $\mathit{Keys}(F) \times \{0,1\}^l \to \{0,1\}^L$ is some function family. Then for any distinguisher A we let

$$\mathbf{Adv}_F^{\mathrm{prf}}(A) \quad = \quad \Pr\left[\, f \stackrel{R}{\leftarrow} F \;:\; A^f = 1 \,\right] - \Pr\left[\, f \stackrel{R}{\leftarrow} \mathrm{Rand}^{l \to L} \;:\; A^f = 1 \,\right] .$$

We associate to F an insecurity function $\mathbf{Adv}_F^{\mathrm{prf}}(\cdot,\cdot)$ defined for any integers $q, t \geq 0$ via

$$\mathbf{Adv}_F^{\mathrm{prf}}(q,t) \quad = \quad \max_A \left\{ \mathbf{Adv}_F^{\mathrm{prf}}(A) \right\} .$$

The maximum is over all distinguishers A that make at most q oracle queries and have "running-time" at most t. ▌

The quantity in quotes needs to be properly defined, and in doing so we adopt some important conventions. First, we define the "execution-time" of A as the time taken for the execution of the experiment $f \stackrel{R}{\leftarrow} F\,;\, b \leftarrow A^f$. Note that we are considering the time for all steps in the experiment, including the time taken to compute replies to oracle queries made by A, and even the time to select a random member of f. (Meaning the time to select a key at random from $\mathit{Keys}(F)$.) The "running-time" of A is defined as the execution time plus the size of the description of A, in our fixed RAM model of computation discussed above.

With F fixed we view $\mathbf{Adv}_F^{\mathrm{prf}}(\cdot,\cdot)$ as a function of q and t. This is the *insecurity function* of F as a PRF, and fully captures the behavior of F as a PRF. It returns the maximum possible advantage that a distinguisher can obtain in telling apart random instances of F from random functions when the distinguisher is restricted to q oracle queries and time t. For any particular values of q, t, the lower this quantity, the better the quality of F as a PRF at the given resource constraints.

Note that under this concrete security paradigm there is no fixed or formal notion of a "secure pseudorandom function family." Every family F simply has some associated insecurity as a PRF. We use the terminology "F is a PRF" only in informal discussions. It is meant to indicate that $\mathbf{Adv}_F^{\mathrm{prf}}(q,t)$ is "low" for "reasonable" values of q, t. Formal result statements will always refer directly to the insecurity function.

PRPs. Luby and Rackoff defined a pseudorandom permutation as a family of permutations that is computationally indistinguishable from the family of random functions [15]. Our notion is a little different. We measure distance from the family of all permutations, not the family of all functions. Note the difference only manifests itself when concrete security is considered. The motivation is that this better models concrete primitives like block ciphers.

Definition 2.2 Suppose F: $\mathit{Keys}(F) \times \{0,1\}^l \rightarrow \{0,1\}^l$ is some function family. Then for any distinguisher A we let

$$\mathbf{Adv}_F^{\mathrm{prp}}(A) \;=\; \Pr\left[f \stackrel{R}{\leftarrow} F \;:\; A^f = 1 \right] - \Pr\left[f \stackrel{R}{\leftarrow} \mathsf{Perm}^l \;:\; A^f = 1 \right] .$$

We associate to F an insecurity function $\mathbf{Adv}_F^{\mathrm{prp}}(\cdot,\cdot)$ defined for any integers $q, t \geq 0$ via

$$\mathbf{Adv}_F^{\mathrm{prp}}(q,t) \;=\; \max_A \; \{ \mathbf{Adv}_F^{\mathrm{prp}}(A) \} .$$

The maximum is over all distinguishers A that make at most q oracle queries and have "running-time" at most t. ▮

The running-time is measured using the same conventions as used for PRFs. Informally, we say that F is a PRP if it is a family of permutations for which $\mathbf{Adv}_F^{\mathrm{prp}}(q,t)$ is "low" for "reasonable" values of q, t.

Where is the key-length? There is one feature of the above parameterizations about which everyone asks. Suppose F is a block cipher with key-length k, meaning $\mathit{Keys}(F) = \{0,1\}^k$. Obviously, the key-length is an important aspect of a block cipher's security. Yet the key-length k does not even appear explicitly in the insecurity function $\mathbf{Adv}_F^{\mathrm{prp}}(q,t)$. Why is this? Because the key-length of F is already reflected in $\mathbf{Adv}_F^{\mathrm{prp}}(q,t)$ to the extent that it matters. The truth is that the key-length itself is not what is of relevance; what matters is the advantage a distinguisher can obtain.

General distance measures. Above we have considered measures of "distance" between a given family F and two particular families, that of all functions and that of all permutations. More generally one can measure the distance between two families of functions.

Definition 2.3 Let F: $\mathit{Keys}(F) \times D \rightarrow R$ and F': $\mathit{Keys}(F') \times D \rightarrow R$ be two function families with a common domain D and a common range R, and let A be a distinguisher. The *advantage* of A is defined as

$$\mathbf{Adv}_{F,F'}^{\mathrm{dist}}(A) \;=\; \Pr\left[f \stackrel{R}{\leftarrow} F \;:\; A^f = 1 \right] - \Pr\left[f \stackrel{R}{\leftarrow} F' \;:\; A^f = 1 \right] .$$

For any integer q we set

$$\mathbf{Adv}_{F,F'}^{\mathrm{dist}}(q) \;=\; \max_A \left\{ \mathbf{Adv}_{F,F'}^{\mathrm{dist}}(A) \right\} .$$

The maximum is over all distinguishers A that make at most q oracle queries. ▮

The above is a "statistical" measure of distance in that it limits only the number of oracle queries and not the running time of the distinguisher. We could define the corresponding "computational" measure by restricting time as well. We don't here, for simplicity, because we will not use that notion in this paper except for the special cases of PRFs and PRPs defined above.

THE BIRTHDAY ATTACK. A few simple facts with regard to the security of PRFs and PRPs are worth noting as they are useful in applying our results and also in getting a better understanding of concrete security. The following says that if E is a block cipher with output-length l then there is an inherent limit to its quality as a PRF, namely that security vanishes as the adversary asks about $2^{l/2}$ queries. This is regardless of the key-size of E, and results only from the fact that E is a family of permutations rather than functions. The reason is the birthday phenomenon. The formal statement is the following.

Proposition 2.4 Let E: $\mathit{Keys}(E) \times \{0,1\}^l \to \{0,1\}^l$ be any family of permutations, and let q be an integer with $1 \le q \le 2^{(l+1)/2}$. Then there is a distinguisher A, making q queries and using $t = O(ql)$ time, such that

$$\mathbf{Adv}_E^{\mathrm{prf}}(A) \;\ge\; 0.316 \cdot \frac{q(q-1)}{2^l} \,.$$

As a consequence

$$\mathbf{Adv}_E^{\mathrm{prf}}(q,t) \;\ge\; 0.316 \cdot \frac{q(q-1)}{2^l} \,.$$

Proof: The distinguisher D is given an oracle for a function g: $\{0,1\}^l \to \{0,1\}^l$. It mounts the following birthday attack:

Distinguisher D^g
  For $i = 1, \ldots, q$ do
    Let x_i be the i-th l-bit string in lexicographic order
    Let $y_i \leftarrow g(x_i)$
  End For
  If $y_1, \ldots, y_q$ are all distinct then return 1, else return 0

To lower bound the advantage of A we claim that

$$\Pr\left[\, f \stackrel{R}{\leftarrow} E \;:\; A^f = 1 \,\right] \;=\; 1 \tag{2}$$

$$\Pr\left[\, f \stackrel{R}{\leftarrow} \mathrm{Rand}^{l \to l} \;:\; A^f = 1 \,\right] \;\le\; 1 - 0.316 \cdot \frac{q(q-1)}{2^l} \,. \tag{3}$$

Equation (2) is clear because if f is an instance of E then f is a permutation and hence $y_1, \ldots, y_q$ are all distinct. Now suppose f is a random function of l-bits to l-bits. Then the probability that $y_1, \ldots, y_q$ are all distinct is $1 - C(2^l, q)$ where $C(2^l, q)$ is the chance of a collision in the experiment of throwing q balls randomly and independently into 2^l buckets. Now Equation (3) follows from Proposition A.1. Subtracting yields the claimed lower bound on the advantage. ▮

PRPs as PRFs. Analyses of constructions (including the CBC-MAC) are often easier assuming the underlying family is a PRF. However, a PRP better models a block cipher. To bridge this gap we often do an analysis assuming the block cipher is a PRF and then use the following to relate the insecurity functions. The following says, roughly, that the birthday attack is the best possible one. A particular family of permutations E may have insecurity in the PRF sense that is greater than that in the PRP sense, but only by an amount of $q^2/2^{l+1}$, the collision probability term in the birthday attack.

Proposition 2.5 Suppose E: $\mathit{Keys}(E) \times \{0,1\}^l \to \{0,1\}^l$ is a family of permutations. Then

$$\mathbf{Adv}_E^{\mathrm{prf}}(q,t) \;\leq\; \mathbf{Adv}_E^{\mathrm{prp}}(q,t) + \frac{q(q-1)}{2^{l+1}} .$$

Proof: Let A by any distinguisher for E versus $\mathrm{Rand}^{l\to l}$ that makes q oracle queries and runs for time at most t. We show that

$$\mathbf{Adv}_E^{\mathrm{prf}}(A) \;\leq\; \mathbf{Adv}_E^{\mathrm{prp}}(A) + \frac{q(q-1)}{2^{l+1}} . \tag{4}$$

The Proposition follows from the definitions of the insecurity functions.

Let $\overline{A}$ denote the distinguisher that first runs A to obtain an output bit b and then returns $\bar{b}$, the complement of b. Let $\Pr_1[\cdot]$ denote the probability that A^f outputs 1 under the experiment $f \stackrel{R}{\leftarrow} \mathrm{Rand}^{l\to l}$, and let $\Pr_2[\cdot]$ denote the probability that A^f outputs 1 under the experiment $f \stackrel{R}{\leftarrow} \mathsf{Perm}^l$. Then

$$\begin{aligned}
\mathbf{Adv}_E^{\mathrm{prf}}(A) &= \Pr\left[f \stackrel{R}{\leftarrow} F : A^f = 1\right] - \Pr_1\left[A^f = 1\right] \\
&= \Pr_1\left[\overline{A}^f = 1\right] - \Pr\left[f \stackrel{R}{\leftarrow} F : \overline{A}^f = 1\right] \\
&= \Pr_1\left[\overline{A}^f = 1\right] - \Pr_2\left[\overline{A}^f = 1\right] + \Pr_2\left[\overline{A}^f = 1\right] - \Pr\left[f \stackrel{R}{\leftarrow} F : \overline{A}^f = 1\right] \\
&= \Pr_1\left[\overline{A}^f = 1\right] - \Pr_2\left[\overline{A}^f = 1\right] + \Pr\left[f \stackrel{R}{\leftarrow} F : A^f = 1\right] - \Pr_2\left[A^f = 1\right] \\
&= \Pr_1\left[\overline{A}^f = 1\right] - \Pr_2\left[\overline{A}^f = 1\right] + \mathbf{Adv}_E^{\mathrm{prp}}(A) .
\end{aligned}$$

So it suffices to show that

$$\Pr_1\left[\overline{A}^f = 1\right] - \Pr_2\left[\overline{A}^f = 1\right] \;\leq\; \frac{q(q-1)}{2^{l+1}} . \tag{5}$$

Assume without loss of generality that all oracle queries of A (they are the same as those of $\overline{A}$) are distinct. Let D denote the event that all the answers are distinct. Then

$$\begin{aligned}
\Pr_1\left[\overline{A}^f = 1\right] &= \Pr_1\left[\overline{A}^f = 1 \mid D\right] \cdot \Pr_1[D] + \Pr_1\left[\overline{A}^f = 1 \mid \neg D\right] \cdot \Pr_1[\neg D] \\
&= \Pr_2\left[\overline{A}^f = 1\right] \cdot \Pr_1[D] + \Pr_1\left[\overline{A}^f = 1 \mid \neg D\right] \cdot \Pr_1[\neg D] \\
&\leq \Pr_2\left[\overline{A}^f = 1\right] + \Pr_1[\neg D] \\
&\leq \Pr_2\left[\overline{A}^f = 1\right] + \frac{q(q-1)}{2^{l+1}} .
\end{aligned}$$

In the last step we used Proposition A.1. This implies Equation (5) and concludes the proof. ▮

It is possible that $\mathbf{Adv}_F^{\mathrm{prp}}(q,t)$ is considerably lower than $\mathbf{Adv}_F^{\mathrm{prf}}(q,t)$ a block cipher F. In particular if F has output-length l then $\mathbf{Adv}_F^{\mathrm{prf}}(q,t)$ becomes substantial by $q = 2^{l/2}$ due to Proposition 2.4, yet it might be that $\mathbf{Adv}_F^{\mathrm{prp}}(q,t)$ stays low even for q far more than $2^{l/2}$. Thus whenever possible we would prefer to bound the insecurity of a block-cipher based construction in terms of the insecurity of the block cipher as PRP. In many cases however (including in this paper) the construction itself is subject to birthday attacks, so it makes little difference in the end.

2.4 Message authentication codes

A message authentication code is a family of functions $MAC\colon\ Keys(MAC) \times Dom(MAC) \to \{0,1\}^s$. The domain is also called the message space; it is the set of strings that can be authenticated using this function. The key a is shared between the sender and the receiver, and when the sender wants to send a message M it computes $\sigma = MAC(a,M)$ and transmits the pair (M,σ) to the receiver. We typically refer to σ as the "MAC" of message M. The receiver re-computes $MAC(a,M)$ and verifies that this equals the value σ that accompanies M.

We define security by adapting and concretizing the notion of security for digital signatures of [12]. An adversary, called a forger in this context, is allowed to mount a chosen-message attack in which it can obtain MACs of messages of its choice. It then outputs a pair (M,σ) and is considered successful if this pair is a valid forgery, meaning that $\sigma = MAC(a,M)$ and furthermore M is "new" in the sense that it was not a message whose MAC the adversary obtained during the chosen-message attack. Formally the success of an adversary A is measured by the following experiment:

Experiment Forge(MAC, A)
  $a \stackrel{R}{\leftarrow} Keys(MAC)$
  $(M,\sigma) \leftarrow A^{MAC(a,\cdot)}$
  If $MAC(a,M) = \sigma$ and M was not a query of A to its oracle
    then return 1 else return 0

The chosen-message attack is captured by giving A an oracle for $MAC(a,\cdot)$. It can invoke this oracle on any message of its choice (with the restriction that this message belongs to the domain $Dom(MAC)$ of the message authentication code) and thereby obtain the MAC of this message. The experiment returns 1 when A is successful in forgery, and 0 otherwise. The output message M must also be in $Dom(MAC)$. In what follows we assume for simplicity that $Dom(MAC) = \{0,1\}^d$ for some integer $d > 0$, since that is the case for the CBC MAC we consider here.

Definition 2.6 Let $MAC\colon\ Keys(MAC) \times \{0,1\}^d \to \{0,1\}^s$ be a message authentication code, and let A be a forger. The success probability of A is defined as

$$\mathbf{Adv}_{MAC}^{\mathrm{mac}}(A) \;=\; \Pr[\,\text{Experiment Forge}(MAC,A) \text{ returns } 1\,]\ .$$

We associate to MAC an insecurity function $\mathbf{Adv}_{MAC}^{\mathrm{mac}}(\cdot,\cdot)$ defined for any integers $q,t \geq 0$ via

$$\mathbf{Adv}_{MAC}^{\mathrm{mac}}(q,t) \;=\; \max_A \left\{\, \mathbf{Adv}_{MAC}^{\mathrm{mac}}(A) \,\right\}\ .$$

The maximum is over all forgers A such that the oracle in Experiment Forge(MAC, A) is invoked at most q times, and the "running time" of A is at most t. ∎

As above, the convention is that resource measures refer to the experiment measuring adversarial success rather than to the adversary itself. In particular, q bounds the total number of queries made in the experiment, meaning that in addition to queries made directly by the adversary, we include in the count the query made to verify the forgery output by the adversary. The running time is the execution time of the experiment (including the time to choose the key and answer oracle queries) plus the size of the description of the adversary.

The definition follows the same format as our previous ones in associating to MAC an insecurity function measuring its quality as a message authentication code.

We have simplified by restricting the domain to strings of a fixed length because that is the case for the basic CBC-MAC we consider. When one wants to discuss MACs over variable-length data, one should augment the set of resources to also consider the total message length, defined as the sum of the lengths of all queries made by the adversary plus the length of the message in the forgery output by the adversary. This quantity becomes an additional input to the insecurity function.

Pseudorandom functions make good message authentication codes. As we remarked in the introduction the reduction is standard [10, 11]. We determine the exact security of this reduction. The following shows that the reduction is almost tight—security hardly degrades at all. This relation means that to prove the security of the CBC MAC as a MAC it is enough to show that the CBC transform preserves pseudorandomness.

Proposition 2.7 Let MAC: $Keys(MAC) \times \{0,1\}^d \to \{0,1\}^s$ be a family of functions, and let $q, t \geq 1$ be integers. Then

$$\mathbf{Adv}^{\mathrm{mac}}_{MAC}(q,t) \;\leq\; \mathbf{Adv}^{\mathrm{prf}}_{MAC}(q,t') + \frac{1}{2^s} \tag{6}$$

where $t' = t + O(s+d)$.

Proof: Let A by any forger attacking the message authentication code MAC. Assume the oracle in Experiment Forge(MAC, A) is invoked at most q times, and the "running time" of A is at most t, these quantities being measured as discussed in Definition 2.6. We design a distinguisher B_A for MAC versus $\mathrm{Rand}^{d\to s}$ such that

$$\mathbf{Adv}^{\mathrm{prf}}_{MAC}(B_A) \;\geq\; \mathbf{Adv}^{\mathrm{mac}}_{MAC}(A) - \frac{1}{2^s}\,. \tag{7}$$

Moreover B will run in time t' and make at most q queries to its oracle, with the time measured as discussed in Definition 2.1. This implies Equation (6) because

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{mac}}_{MAC}(q,t) &= \max_A \{\, \mathbf{Adv}^{\mathrm{mac}}_{MAC}(A) \,\} \\
&\leq \max_A \left\{\, \mathbf{Adv}^{\mathrm{prf}}_{MAC}(B_A) + 2^{-s} \,\right\} \\
&= \max_A \left\{\, \mathbf{Adv}^{\mathrm{prf}}_{MAC}(B_A) \,\right\} + 2^{-s}
\end{aligned}$$

$$
\begin{aligned}
&\leq \max_{B} \left\{ \mathbf{Adv}^{\mathrm{prf}}_{MAC}(B) \right\} + 2^{-s} \\
&= \mathbf{Adv}^{\mathrm{prf}}_{MAC}(q, t') + 2^{-s} \,.
\end{aligned}
$$

Above the first equality is by the definition of the insecurity function in Definition 2.6. The following inequality uses Equation (7). Next we simplify using properties of the maximum, and conclude by using the definition of the insecurity function as per Definition 2.1.

So it remains to design B_A such that Equation (7) is true. Remember that B_A is given an oracle for a function $f\colon \{0,1\}^d \to \{0,1\}^s$. It will run A, providing it an environment in which A's oracle queries are answered by B_A. When A finally outputs its forgery, B_A checks whether it is correct, and if so bets that f must have been an instance of the family MAC rather than a random function.

By assumption the oracle in Experiment Forge(MAC, A) is invoked at most q times, and for simplicity we assume it is exactly q. This means that the number of queries made by A to its oracle is $q-1$. Here now is the code implementing B_A.

Distinguisher B_A^f
    For $i = 1, \ldots, q-1$ do
        When A asks its oracle some query, M_i, answer with $f(M_i)$
    End For
    A outputs (M, σ)
    $\sigma' \leftarrow f(M)$
    If $\sigma = \sigma'$ and $M \notin \{M_1, \ldots, M_{q-1}\}$
        then return 1 else return 0

Here B_A initializes A with some random sequence of coins and starts running it. When A makes its first oracle query M_1, algorithm B_A pauses and computes $f(M_1)$ using its own oracle f. The value $f(M_1)$ is returned to A and the execution of the latter continues in this way until all its oracle queries are answered. Now A will output its forgery (M, σ). B_A verifies the forgery, and if it is correct, returns 1.

We now proceed to the analysis. We claim that

$$
\Pr\left[f \stackrel{R}{\leftarrow} MAC \;:\; B_A^f = 1 \right] = \mathbf{Adv}^{\mathrm{mac}}_{MAC}(A) \tag{8}
$$

$$
\Pr\left[f \stackrel{R}{\leftarrow} \mathrm{Rand}^{d \to s} \;:\; B_A^f = 1 \right] \leq \frac{1}{2^s} \,. \tag{9}
$$

Subtracting, we get Equation (7), and from the code it is evident that B_A makes q oracle queries. Taking into account our conventions about the running times referring to that of the entire experiment it is also true that the running time of B_A is $t + O(d+s)$. So it remains to justify the two equations above.

In the first case f is an instance of MAC, so that the simulated environment that B_A is providing for A is exactly that of experiment Forge(MAC, A). Since B_A returns 1 exactly when A makes a successful forgery, we have Equation (8).

In the second case, A is running in an environment that is alien to it, namely one where a random function is being used to compute MACs. We have no idea what A will do in this environment, but no matter what, we know that the probability that $\sigma = f(M)$ is 2^{-s}, because f is a random function, as long as A did not query M of its oracle. Equation (9) follows. ▮

2.5 The CBC transform

Let $f\colon \{0,1\}^l \to \{0,1\}^l$ be a function and $m > 0$ an integer. We associate to them another function $f^{(m)}\colon \{0,1\}^{ml} \to \{0,1\}^l$ called the CBC-MAC of f, as follows. If x is a string in the domain $\{0,1\}^{ml}$ then we view it as a sequence of l-bit blocks and let x_i denote the i-th block for $i = 1, \ldots, m$. We then set

Function $f^{(m)}(x_1 \cdots x_m)$
    $y_0 \leftarrow 0^l$
    For $i = 1, \ldots, m$ do $y_i \leftarrow f(y_{i-1} \oplus x_i)$
    Return y_m

The construct extends in a natural way to a family of functions. The *CBC transform* associates to a given family $F\colon \mathit{Keys}(F) \times \{0,1\}^l \to \{0,1\}^l$ and integer $m > 0$ another family that we denote $\mathsf{CBC}^m\text{-}F$. It maps $\mathit{Keys}(F) \times \{0,1\}^{ml} \to \{0,1\}^l$. As the notation indicates, the new family has the same key-space and range as F, but a larger domain. For any key $a \in \mathit{Keys}(F)$ and any input $x = x_1 \ldots x_m \in \{0,1\}^{ml}$ we set

$$\mathsf{CBC}^m\text{-}F_a(x_1 \ldots x_m) = F_a^{(m)}(x_1 \cdots x_m) \; .$$

Or, in more detail

Function $\mathsf{CBC}^m\text{-}F(a, x_1 \ldots x_m)$
    $y_0 \leftarrow 0^l$
    For $i = 1, \ldots, m$ do $y_i \leftarrow F(a, y_{i-1} \oplus x_i)$
    Return y_m

We stress that the domain of $\mathsf{CBC}^m\text{-}F$ consists of strings of length exactly ml, not at most ml.

3 Pseudorandomness of the CBC-MAC

In this section we show that the CBC-MAC transform applied to a PRF yields a PRF, and assess how the security of the transformed family depends on that of the given family. In the next section we derive the implications on the security of the CBC-MAC as a message authentication code.

The practical concern is the case where the family to which the CBC-MAC transform is applied is a block cipher. The analysis however begins by considering a "thought-experiment". Namely, we consider the CBC-MAC of a random function, or, more formally, the family resulting from applying the CBC-MAC transform to the family $\mathrm{Rand}^{l \to l}$ of all functions of l-bits to l-bits. By considering this, we are asking whether the CBC-MAC transform has any inherent weaknesses, meaning weaknesses that would exist even for "ideal" block ciphers. Our first result below (the information-theoretic case of the CBC theorem) says that at least in a qualitative sense, the answer is no: the transform is provably secure. The theorem goes further, providing also a quantitative bound on the insecurity. The second result (the computational case of the CBC theorem) considers the case of a given block cipher or family F, and bounds the insecurity of the transformed family in terms of that of the original one.

Below we first present and discuss both theorems. We then prove the second, which follows from the first by relatively standard means. We then go on to the main technical part of the paper, which is the proof of the information theoretic case of the CBC theorem.

3.1 Main results

INFORMATION THEORETIC CASE. The information-theoretic case of the CBC theorem considers an adversary A of unrestricted computational power. She is faced with the following problem. She is given an oracle to a function g chosen in one of the following ways: either g is a random function of ml bits to l bits, or $g = f^{(m)}$ for a random function f of l bits to l bits. The choice between these two possibilities is made according to a hidden coin flip. What is A's advantage in figuring out which type of oracle she has, as a function of the number q of oracle queries she makes? The formal statement is made in terms of the distance function of Definition 2.3. The proof of the following is in Section 3.3.

Theorem 3.1 [CBC Theorem: Information-Theoretic Case] Let $l, m \geq 1$ and $q \geq 0$ be integers. Let $C = \mathsf{CBC}^m\text{-Rand}^{l \to l}$ and let $R = \text{Rand}^{ml \to l}$. Then

$$\mathbf{Adv}^{\text{dist}}_{C,R}(q) \;\leq\; 1.5 \cdot \frac{q^2 m^2}{2^l} \, .$$

In other words an adversary making q queries cannot have an advantage exceeding $3q^2m^2 \cdot 2^{-l-1}$, no matter what strategy this adversary uses.

NUMERICAL EXAMPLE. Suppose $l = 128$ bits and we use the CBC MAC over a random function to authenticate $q = 2^{30}$ messages of 16 KBytes each (so $m = 2^{10}$ blocks). Then no adversary, after adaptively obtaining the MACs of $q = 2^{30}$ strings, has an advantage as large as 5.4×10^{-15} at distinguishing these MACs from purely random strings. Consequently, no adversary will be able, after having performed the above tests, to forge one new MAC with probability as large as 5.4^{-15}. (The two probabilities differ by an additive factor of 2^{-128}, as per Proposition 2.7).)

COMPUTATIONAL CASE. Now suppose F: $\mathit{Keys}(F) \times \{0,1\}^l \to \{0,1\}^l$ is some given family of functions, for example a block cipher like DES or RC6. Assume the given family is a PRF (or PRP). We want to know how secure is the CBC-MAC based on this family. The following theorem considers an adversary A given an oracle to a function g chosen in one of the following ways: either g is a random function of ml bits to l bits, or $g = f^{(m)}$ for a random instance f of family F. The choice between these two possibilities is made according to a hidden coin flip. What is A's advantage in figuring out which type of oracle she has, as a function of the number q of oracle queries she makes and the amount t of computation time she uses? The proof of the following is in Section 3.2.

Theorem 3.2 [CBC Theorem: Computational Case] Let $l, m \geq 1$ and $q, t \geq 0$ be integers. Let F: $\mathit{Keys}(F) \times \{0,1\}^l \to \{0,1\}^l$ be a family of functions. Then

$$\mathbf{Adv}^{\text{prf}}_{\mathsf{CBC}^m\text{-}F}(q,t) \;\leq\; \mathbf{Adv}^{\text{prf}}_{F}(q',t') + 1.5 \cdot \frac{q^2 m^2}{2^l} \tag{10}$$

$$\leq\; \mathbf{Adv}^{\text{prp}}_{F}(q',t') + \frac{q^2 m^2}{2^{l-1}} \tag{11}$$

where $q' = mq$ and $t' = t + O(mql)$.

The constant hidden in the O-notation depends only on details of the model of computation. One should think of t as being large compared to the other parameters, so that $t' \approx t$.

Here is one way to interpret the theorem. Suppose F was $\mathrm{Rand}^{l\to l}$, the family of all functions. Then Theorem 3.1 says that $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CBC}^m\text{-}F}(q,t)$ would be at most $1.5 \cdot q^2 m^2 \cdot 2^{-l}$. Theorem 3.2 says that when F is not $\mathrm{Rand}^{l\to l}$ we need to just add in the "distance" of F to $\mathrm{Rand}^{l\to l}$, meaning $\mathbf{Adv}^{\mathrm{prf}}_{F}(q',t')$. Viewed in this way, Theorem 3.2 is quite intuitive.

The reason for the second inequality in Theorem 3.2 is that block ciphers are more naturally viewed as PRPs and thus it is more useful to phrase the bound in terms of the insecurity of F as a PRP. (However, in this case there is little numerical difference in the two because the second term in the first inequality is already proportional to q^2.)

Numerical example. Suppose we use the CBC MAC where the underlying block cipher is the AES algorithm (a block cipher soon to be selected by the National Institute of Standards). This block cipher will have a block-length of $l = 128$ bits. Suppose some scientist finds a practical method which, after obtaining the MACs of $q = 2^{30}$ messages, each message of length 16 KBytes, distinguishes with advantage of 1% the 2^{30} answers just obtained and 2^{30} random strings. Then there is some equally practical method—it's running time is essentially that of the first method—which has advantage of at least $0.01 - 7.2 \times 10^{-15} \approx 1\%$ at differentiating AES values on 2^{40} points from as many random distinct points. This would be bad news for the AES, and might be seen as highly unlikely for a modern and conservatively designed block cipher.

3.2 Proof of Theorem 3.2

In the asymptotic setting such a proof would normally proceed by contradiction. We would assume there was an adversary A that succeeded in "breaking" CBC^m-F, and then design another adversary B_A that succeeds in "breaking" F while using resources polynomial in those used by the original adversary. The underlying idea remains the same, but in the concrete security setting it less convenient to use contradiction. We simply associate to any A some B_A and then proceed to relate their success probabilities. The relations must be done carefully and with regard to tightness of the analysis. The proof below will use Theorem 3.1 in a crucial way. The main lemma below equates the advantage of A in distinguishing between CBC^m-F and $\mathrm{Rand}^{ml\to l}$ with the sum of two other advantages. The first is that obtained by B_A in distinguishing between F and $\mathrm{Rand}^{l\to l}$, while the second is that of A in distinguishing between CBC^m-$\mathrm{Rand}^{l\to l}$ and $\mathrm{Rand}^{ml\to l}$.

Lemma 3.3 Let A be a distinguisher for CBC^m-F versus $\mathrm{Rand}^{ml\to l}$ which makes at most q oracle queries and has running time at most t. Then there is a distinguisher B_A for F versus $\mathrm{Rand}^{l\to l}$ such that

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CBC}^m\text{-}F}(A) \quad = \quad \mathbf{Adv}^{\mathrm{prf}}_{F}(B_A) + \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CBC}^m\text{-}\mathrm{Rand}^{l\to l}}(A) \ ,$$

and, furthermore, B_A makes at most q' oracle queries and has running time at most t', where $q' = mq$ and $t' = t + O(mql)$.

We conclude the proof of Theorem 3.2 given the above lemma and then return to prove the lemma.

Proof of Theorem 3.2: Let A be a distinguisher for $\mathsf{CBC}^m\text{-}F$ versus $\mathrm{Rand}^{ml\to l}$ which makes at most q oracle queries and has running time at most t. Then

$$\begin{aligned}\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CBC}^m\text{-}\mathrm{Rand}^{l\to l}}(A) &\leq \mathbf{Adv}^{\mathrm{dist}}_{\mathsf{CBC}^m\text{-}\mathrm{Rand}^{l\to l},\mathrm{Rand}^{ml\to l}}(q) \\ &\leq 1.5\cdot\frac{q^2m^2}{2^l}\,.\end{aligned}$$

Here the first inequality is true because A makes at most q queries, and the second inequality is by Theorem 3.1. Let B_A be as given by Lemma 3.3. By that lemma and the above we have

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CBC}^m\text{-}F}(A) \;\leq\; \mathbf{Adv}^{\mathrm{prf}}_{F}(B_A)+1.5\cdot\frac{q^2m^2}{2^l}\,. \tag{12}$$

Furthermore we know that B_A makes at most q' oracle queries and has running time at most t'. Equation (10) follows because

$$\begin{aligned}\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CBC}^m\text{-}F}(q,t) &= \max_A\left\{\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CBC}^m\text{-}F}(A)\right\} \\ &\leq \max_A\left\{\mathbf{Adv}^{\mathrm{prf}}_{F}(B_A)+1.5\cdot\frac{q^2m^2}{2^l}\right\} \\ &= \max_A\left\{\mathbf{Adv}^{\mathrm{prf}}_{F}(B_A)\right\}+1.5\cdot\frac{q^2m^2}{2^l} \\ &\leq \max_B\left\{\mathbf{Adv}^{\mathrm{prf}}_{F}(B)\right\}+1.5\cdot\frac{q^2m^2}{2^l} \\ &= \mathbf{Adv}^{\mathrm{prf}}_{F}(q',t')+1.5\cdot\frac{q^2m^2}{2^l}\,.\end{aligned}$$

Above the first equality is by the definition of the insecurity function. The following inequality uses Equation (12). Next we simplify using properties of the maximum, and conclude by again using the definition of the insecurity function.

Equation (11) now follows from Equation (10) and Proposition 2.5:

$$\begin{aligned}\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CBC}^m\text{-}F}(q,t) &\leq \mathbf{Adv}^{\mathrm{prf}}_{F}(q',t')+1.5\cdot\frac{q^2m^2}{2^l} \\ &\leq \mathbf{Adv}^{\mathrm{prp}}_{F}(q',t')+1.5\cdot\frac{q^2m^2}{2^l}+0.5\cdot\frac{q'(q'-1)}{2^l} \\ &\leq \mathbf{Adv}^{\mathrm{prp}}_{F}(q',t')+\frac{q^2m^2}{2^{l-1}}\,.\end{aligned}$$

The last inequality upper bounds $q'-1$ by q' and substitutes $q'=qm$. ▮

Proof of Lemma 3.3: Distinguisher B_A gets an oracle $f\colon \{0,1\}^l \to \{0,1\}^l$. It will run A as a subroutine, using f to simulate the oracle $g\colon \{0,1\}^{ml} \to \{0,1\}^l$ that A expects. That is, B_A will itself provide the answers to A's oracle queries by appropriately using f.

Distinguisher B_A^f
For $i = 1, \ldots, q$ do
When A asks its oracle some query, M_i, answer with $f^{(m)}(M_i)$
End For
A outputs a bit, b
Return b

Here B_A initializes A with some random sequence of coins and starts running it. When A makes its first oracle query M_1, algorithm B_A pauses and computes $f^{(m)}(M_1)$, which it can do because $f^{(m)}(\cdot)$ can be computed by making m calls to $f(\cdot)$. The value $f^{(m)}(M_1)$ is returned to A and the execution of the latter continues in this way until all its oracle queries are answered. Now A will output its guess bit b. Adversary B_A simply returns the same as its own guess bit.

We now proceed to the analysis. The oracle supplied to A by B_A in the simulation is $f^{(m)}$ where f is B_A's oracle, and hence

$$\begin{aligned}\mathbf{Adv}_F^{\mathrm{prf}}(B_A) &= \Pr\left[f \stackrel{R}{\leftarrow} F \,:\, B_A^f = 1\right] - \Pr\left[f \stackrel{R}{\leftarrow} \mathrm{Rand}^{l \to l} \,:\, B_A^f = 1\right] \\ &= \Pr\left[g \stackrel{R}{\leftarrow} \mathsf{CBC}^m\text{-}F \,:\, A^g = 1\right] - \Pr\left[g \stackrel{R}{\leftarrow} \mathsf{CBC}^m\text{-}\mathrm{Rand}^{l \to l} \,:\, A^g = 1\right] .\end{aligned}$$

On the other hand

$$\begin{aligned}&\mathbf{Adv}_{\mathsf{CBC}^m\text{-}\mathrm{Rand}^{l\to l}}^{\mathrm{prf}}(A) \\ &= \Pr\left[g \stackrel{R}{\leftarrow} \mathsf{CBC}^m\text{-}\mathrm{Rand}^{l \to l} \,:\, A^g = 1\right] - \Pr\left[g \stackrel{R}{\leftarrow} \mathrm{Rand}^{ml \to l} \,:\, A^g = 1\right] .\end{aligned}$$

Take the sum of the two equations above, and exploit the cancellation to get

$$\Pr\left[g \stackrel{R}{\leftarrow} \mathsf{CBC}^m\text{-}F \,:\, A^g = 1\right] - \Pr\left[g \stackrel{R}{\leftarrow} \mathrm{Rand}^{ml \to l} \,:\, A^g = 1\right] .$$

But this is exactly $\mathbf{Adv}_{\mathsf{CBC}^m\text{-}F}^{\mathrm{prf}}(A)$. ▮

3.3 Proof of Theorem 3.1

Intuition. Before going into the formal proof, we give some intuition for it, with the caveat that this theorem seems prone to intuitive arguments that don't hold up to more rigorous scrutiny.

First, we consider what can go wrong. In computing MACs of the form $f^{(m)}(x_1 \cdots x_m)$, we compute quantities of the form $f^{(i)}(x_1 \cdots x_i)$ and $f^{(i)}(x_1 \cdots x_i) \oplus x_{i+1}$. Even though the results of these subcomputations are hidden from the adversary, certain coincidences, which we call *collisions*, allow the adversary to distinguish $f^{(m)}$ from a truly random function. For example, suppose that for unequal sequences of i blocks, $x_1 \cdots x_i$ and $x'_1 \cdots x'_i$,

$$f^{(i)}(x_1 \cdots x_i) = f^{(i)}(x'_1 \cdots x'_i).$$

Then $f^{(m)}(x_1 \cdots x_i x_{i+1} \cdots x_m) = f^{(m)}(x'_1 \cdots x'_i x_{i+1} \cdots x_m)$ for any $x_{i+1} \cdots x_m$, a clear deviation from randomness. We therefore give up if such collisions ever occur. Indeed, much of our proof

(Lemma 3.8, and the supporting machinery of Lemmas 3.6 and 3.7) is spent showing that these collisions occur only rarely regardless of the strategy used by the adversary.

On the positive side, we have the following simple observation: if $f(x)$ hasn't been computed before or constrained in any way, then its value is uniformly distributed over l-bit strings, independent of any previous computations. This is good in a direct way, because we want the values of the MAC to be random. It is also good in an indirect way, because such a random value is highly unlikely to cause a collision. The rough intuition is that we start in a collision-free state, so any new values generated will be random. Since these values are random, they are unlikely to form a collision.

We represent all possible subcomputations that might be performed as nodes in a large tree; the subcomputations induced by the adversary form a subtree. The specific values of $f^{(i)}(x_1 \cdots x_i)$ and $f^{(i)}(x_1 \cdots x_i) \oplus x_{i+1}$ are represented as labels (Z and Y, respectively) of the nodes in this tree.

Our proof relies on the fact that the adversary has only a partial view of the subcomputations performed on its behalf; if it were omniscient it could easily generate a collision. We therefore take a Baysean viewpoint. Given the adversary's current knowledge, the "hidden" subcomputations have some induced conditional probability distribution. We show that despite the adversary's view, the values of these subcomputations are sufficiently random that our intuition that "random outputs don't cause collisions" is valid.

SETUP. We now begin the full proof. Fix an adversary A. Since we are not restricting computation time a standard argument shows that we may assume without loss of generality that A is deterministic. We begin with some definitions. The connection between these definitions and the game we are considering will be made later.

QUERY SEQUENCES AND LABELINGS. Call the 2^l-ary rooted tree of depth m the *full* tree. A sequence $X = x_1 \cdots x_i$ of l-bit strings ($1 \le i \le m$) names a node at depth i in the natural way. The root is denoted Λ. A function $f\colon \{0,1\}^l \to \{0,1\}^l$ induces labelings $Y_f(X)$ and $Z_f(X)$, recursively defined by

$$\begin{aligned} Z_f(\Lambda) &= 0^l, \\ Y_f(x_1 \cdots x_i) &= Z_f(x_1 \cdots x_{i-1}) \oplus x_i \quad \text{and} \\ Z_f(X) &= f(Y_f(X)) \qquad \text{for } X \neq \Lambda. \end{aligned}$$

$Y_f(\Lambda)$ is undefined. We drop the subscript when f is clear from context or unimportant to the discussion. We sometimes refer to Y as the *input labeling* and Z as the *output labeling*, motivated by the relation $Z(X) = f(Y(X))$. Note that for $X \neq \Lambda$, $Z(X) = f^{(i)}(X)$. We sometimes use *labeling* to refer to both labelings.

A sequence of distinct non-root nodes $X_1, \ldots, X_n$ is a *query sequence* if for every i there is a $j < i$ such that the parent of X_i is either X_j or Λ. The *query tree* associated to a query sequence $X_1, \ldots, X_n$ is the (rooted) subtree of the full tree induced by the nodes $\{\Lambda, X_1, \ldots, X_n\}$; it consists of a collection of root emanating paths. Nodes at depth m are called *border* nodes.

Definition 3.4 A labeling Y of $X_1, \ldots, X_n$ is *collision free* if $Y(X_1), \ldots, Y(X_n)$ are distinct; collision-freeness for Z is defined analogously.

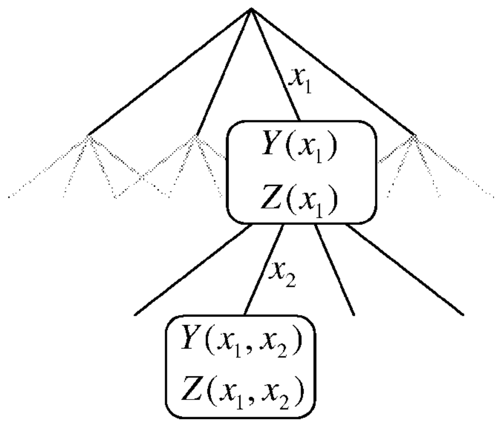

Figure 1: Querying the value of $f^{(m)}(X)$ for $X = (x_1, x_2)$ corresponds to traversing a path from the root to a leaf in the full tree. This path follows the edge x_1 to node x_1, labeled with $Y(x_1) = x_1$ and $Z(x_1) = f(x_1)$. The path then follows the edge x_2 to node (x_1, x_2), labeled with $Y(x_1, x_2) = Z(x_1) \oplus x_2$ and $Z(x_1, x_2) = f(Y(x_1, x_2))$. The adversary only learns $Z(x_1, x_2) = f^{(m)}(X)$.

Definition 3.5 A *border labeling* $\hat{Z}$ of a query sequence is a map assigning an l-bit string to each border node in the query tree. A labeling Z is consistent with a border labeling $\hat{Z}$ if the two agree on the border nodes.

A NEW VIEW OF THE GAME. A query $x_1 \cdots x_m$ of the adversary to the g-oracle can be thought of as specifying a root to border path in the full tree. Now imagine a slightly different game in which the adversary has more power. She can sequentially make qm queries, each a node in the full tree, with the restriction that her queries form a query sequence $X_1, \ldots, X_{qm}$. She receives no answer to queries which are internal nodes of the full tree, but when she queries a border node she receives its Z_f value. It suffices to prove the theorem for adversaries with this enlarged set of capabilities.

THE BASIC RANDOM VARIABLES. The query sequence, its Z_f-labeling, and the values returned to the adversary are all random variables over the random choice of $f \in \mathrm{Rand}^{l \to l}$. We denote by $\mathsf{X}_1, \ldots, \mathsf{X}_{qm}$ the random variables which are the queries of A. We denote by Z_n the labeling of $\mathsf{X}_1, \ldots, \mathsf{X}_n$ specified by Z_f. We denote by $\hat{\mathsf{Z}}_n$ the labeling of the border nodes of the query tree associated to $\mathsf{X}_1, \ldots, \mathsf{X}_n$ specified by Z_f. The input labeling induced by Z_n is denoted Y_n. The *view* of A after her n-th query is the random variable $\mathsf{View}_n = (\mathsf{X}_1, \ldots, \mathsf{X}_n \,;\, \hat{\mathsf{Z}}_n)$.

EQUI-PROBABILITY OF COLLISION-FREE LABELINGS. The following lemma fixes the number n of queries that A has made. It then fixes a particular view $(X_1, \ldots, X_n \,;\, \hat{Z})$ of A. It now examines the distribution on labelings from the point of view of A. It says that as far as A can tell, all collision free labelings of $X_1, \ldots, X_n$ consistent with her current view are equally likely.

Lemma 3.6 Let $1 \le n \le qm$ and let $X_1, \ldots, X_n$ be a query sequence. Let Z_n^1 and Z_n^2 be collision free (output) labelings of $X_1, \ldots, X_n$ which are consistent with a border labeling $\hat{Z}_n$ of $X_1, \ldots, X_n$. Then

$$\Pr\left[\, \mathsf{Z}_n = Z_n^1 \mid \mathsf{View}_n = (X_1, \ldots, X_n \,;\, \hat{Z}_n) \,\right] = \Pr\left[\, \mathsf{Z}_n = Z_n^2 \mid \mathsf{View}_n = (X_1, \ldots, X_n \,;\, \hat{Z}_n) \,\right],$$

where the probability is taken over the choice of f.

Proof of Lemma 3.6: The proof is by induction on n. The lemma holds vacuously for $n = 1$. Assuming the lemma for $1, \ldots, n-1$ we now prove it for n. Let Z_{n-1}^i be the restriction of Z_n^i to $X_1, \ldots, X_{n-1}$ $(i = 1, 2)$. Let $\hat{Z}_{n-1}$ be the restriction of $\hat{Z}_n$ to the border nodes of $X_1, \ldots, X_{n-1}$. Let V_i be the event $\mathsf{View}_i = (X_1, \ldots, X_i \,;\, \hat{Z}_i)$ and let $\Pr_j[\cdot] = \Pr[\,\cdot \mid V_j\,]$, for $j = n-1, n$. Let Y_j^i denote the input labeling induced by Z_j^i for $j = n-1, n$ and $i = 1, 2$. We consider two cases.

Case 1. X_n is not a border node.

For $i = 1, 2$ we have:

$$\begin{aligned}
\Pr_n\left[\, \mathsf{Z}_n = Z_n^i \,\right] &= \Pr_{n-1}\left[\, \mathsf{Z}_n = Z_n^i \,\right] \\
&= \Pr_{n-1}\left[\, \mathsf{Z}_{n-1} = Z_{n-1}^i \,\right] \cdot \Pr_{n-1}\left[\, \mathsf{Z}_n(X_n) = Z_n^i(X_n) \mid \mathsf{Z}_{n-1} = Z_{n-1}^i \,\right] \\
&= \Pr_{n-1}\left[\, \mathsf{Z}_{n-1} = Z_{n-1}^i \,\right] \cdot 2^{-l} .
\end{aligned}$$

The proof is concluded by using the inductive hypothesis. We now justify the above three equalities. Since X_n is not a border node, it is determined by $X_1, \ldots, X_{n-1} \,;\, \hat{Z}_{n-1}$. This means that $\Pr_n[\,\cdot\,]$

equals $\Pr_{n-1}[\,\cdot\,]$ which justifies the first equality. The second is just conditioning. Since Z_n^i is collision free, Y_n^i differs from all the points $Y_{n-1}^i(X_1),\ldots,Y_{n-1}^i(X_{n-1})$ on which the underlying randomly chosen f has been evaluated so far. But $\mathsf{Z}_n(X_n) = f(Y_n^i(X_n))$, so the second term in the product in the second line above is indeed 2^{-l}, implying the third equality. Note that the above probabilities are *not* conditioned on Z_n being collision free.

Case 2. X_n is a border node.

Both Z_n^1 and Z_n^2 are by assumption consistent with $\hat{Z}_n$. But since X_n is a border node, the value $\hat{\zeta} \stackrel{\text{def}}{=} \mathsf{Z}_n(X_n)$ is contained in $\hat{Z}_n$, and $\hat{\zeta} = Z_n^1(X_n) = Z_n^2(X_n)$. Now for $i = 1, 2$ we have:

$$
\begin{aligned}
\Pr_n\left[\mathsf{Z}_{n-1} = Z_{n-1}^i\right] &= \Pr_{n-1}\left[\mathsf{Z}_{n-1} = Z_{n-1}^i \mid \mathsf{Z}_n(X_n) = \hat{\zeta}\right] \\
&= \Pr_{n-1}\left[\mathsf{Z}_n(X_n) = \hat{\zeta} \mid \mathsf{Z}_{n-1} = Z_{n-1}^i\right] \cdot \frac{\Pr_{n-1}\left[\mathsf{Z}_{n-1} = Z_{n-1}^i\right]}{\Pr_{n-1}\left[\mathsf{Z}_n(X_n) = \hat{\zeta}\right]} \\
&= 2^{-l} \cdot \frac{\Pr_{n-1}\left[\mathsf{Z}_{n-1} = Z_{n-1}^i\right]}{\Pr_{n-1}\left[\mathsf{Z}_n(X_n) = \hat{\zeta}\right]} .
\end{aligned}
$$

The events V_n and $V_{n-1} \wedge (\mathsf{Z}_n(X_n) = \hat{\zeta})$ are the same, since $\mathsf{Z}_n(X_n) = \hat{\zeta}$ "fills" in the portion of View_n not contained in View_{n-1}; the first equality follows. The second equality follows from Bayes rule. That the first term of the product in the second line above is indeed 2^{-l} is argued as in Case 1 based on the fact that Z_n^i is collision free. Now note the denominator in the fraction above is independent of $i \in \{1, 2\}$. Applying the inductive hypothesis, we conclude that

$$
\Pr_n\left[\mathsf{Z}_{n-1} = Z_{n-1}^1\right] = \Pr_n\left[\mathsf{Z}_{n-1} = Z_{n-1}^2\right] . \tag{13}
$$

Now for $i = 1, 2$:

$$
\begin{aligned}
\Pr_n\left[\mathsf{Z}_n = Z_n^i\right] &= \Pr_n\left[\mathsf{Z}_{n-1} = Z_{n-1}^i\right] \cdot \Pr_n\left[\mathsf{Z}_n(X_n) = \hat{\zeta} \mid \mathsf{Z}_{n-1} = Z_{n-1}^i\right] \\
&= \Pr_n\left[\mathsf{Z}_{n-1} = Z_{n-1}^i\right] \cdot 1 .
\end{aligned}
$$

The second term in the above product is 1 because V_n contains $\hat{\zeta}$ as the value of $\mathsf{Z}_n(X_n)$. The proof for this case is concluded by applying Equation 13. ∎

More definitions. Let $X_1,\ldots,X_n$ be a query sequence. We will discuss labelings $\vec{z}$ which assign values only to some specified subset S of this sequence. The input labeling induced by $\vec{z}$ assigns values to all nodes of $X_1,\ldots,X_n$ which are at level one and all nodes whose parents are in S. We can discuss collision freeness of such labelings, or their consistency with a border labeling, in the usual way. We denote by Z_n^S the labeling of S given by restricting Z_n to S. Let $\mathsf{ColFree}(Z)$ be true if labeling Z is collision free.

Unpredictability of internal labels. The following lemma fixes the number n of queries that A has made, as well as a particular view $X_1,\ldots,X_n\,;\,\hat{Z}$ of A. It now makes the assumption that the current labeling Z_n is collision free; think of this fact as being known to A. Given all this, it examines the distribution on labels from the point of view of A. Some labels are known:

for example, the Z_n values of border nodes and the Y_n values of nodes at depth one. The lemma says that all other labels are essentially unpredictable. First, it considers a node $x_1 \cdots x_i x_{i+1}$ that is at depth at least two, and says that even given the output labels (i.e. Z_n values) of all nodes except its parent $x_1 \cdots x_i$, the Y_n value of $x_1 \cdots x_i x_{i+1}$ is almost uniformly distributed. Second, it considers a node $x_1 \cdots x_i$ that is not a border node, and says that even given the output labels of all other nodes, the Z_n value of $x_1 \cdots x_i$ is almost uniformly distributed. For technical reasons the lemma requires a bound on the number n of queries that have been made.

Lemma 3.7 Let $1 \le n \le qm-1$ and suppose $n^2/4 + n - 1 \le 2^l/2$. Let $X_1, \ldots, X_n$ be a query sequence and let $\hat{Z}$ be a labeling of the border nodes of $X_1, \ldots, X_n$. Let

$$\Pr_n[\cdot] \;=\; \Pr\Big[\cdot \mid \mathsf{View}_n = (X_1, \ldots, X_n \,;\, \hat{Z}) \wedge \mathsf{ColFree}(\mathsf{Z}_n)\Big] \;,$$

where the probability is taken over the choice of f. Suppose $x_1 \cdots x_i \in \{X_1, \ldots, X_n\}$ is a non-border node and let $S = \{X_1, \ldots, X_n\} - \{x_1 \cdots x_i\}$. Suppose $\vec{z}$: $S \to \{0,1\}^l$ is a collision free labeling of S that is consistent with $\hat{Z}$.

(1) Let $x_1 \cdots x_i x_{i+1} \in S$ be a child of $x_1 \cdots x_i$. Then for any $y^* \in \{0,1\}^l$:

$$\Pr_n\Big[\mathsf{Y}_n(x_1 \cdots x_i x_{i+1}) = y^* \mid \mathsf{Z}_n^S = \vec{z}\Big] \;\le\; 2 \cdot 2^{-l} \,.$$

(2) For any $z^* \in \{0,1\}^l$:

$$\Pr_n\Big[\mathsf{Z}_n(x_1 \cdots x_i) = z^* \mid \mathsf{Z}_n^S = \vec{z}\Big] \;\le\; 2 \cdot 2^{-l} \,.$$

Proof of Lemma 3.7: Let $x_1 \cdots x_i x_{i+1}^u$ $(u = 1, \cdots, s)$ be the children of $x_1 \cdots x_i$. Denote by $\textsc{children}(x_1 \cdots x_i)$ the set $\{x_1 \cdots x_i x_{i+1}^1, \cdots, x_1 \cdots x_i x_{i+1}^s\}$.Let $\vec{y}$: $\{X_1, \ldots, X_n\} - \textsc{children}(x_1 \cdots x_i) \to \{0,1\}^l$ be the input labeling induced by $\vec{z}$. We prove the two claims in turn.

Proof of **(1)**. Let's begin by giving some intuition for the proof. We observe that with $\vec{z}$ given, if we assign an input label $y \in \{0,1\}^l$ to $x_1 \cdots x_i x_{i+1}$ then the value of Z_n at the parent node $x_1 \cdots x_i$ is determined; given this, the values of Y_n at the other children of $x_1 \cdots x_i$ are also determined. Thus, both Z_n and Y_n are now fully determined for all nodes $X_1, \ldots, X_n$. We will show that there is a large set $S(\vec{z})$ of these y values for which the determined labeling is collision free. Moreover, all collision free labelings have this form and are equally likely by Lemma 3.6; thus as far as A can tell, the value at $x_1 \cdots x_i x_{i+1}$ is equally likely to be anything from the set $S(\vec{z})$. The formal proof follows.

Assume without loss of generality that $x_1 \cdots x_i x_{i+1}^1 = x_1 \cdots x_i x_{i+1}$. Let $y \in \{0,1\}^l$ be some fixed string. Now define the labeling $Z_{\vec{z},y}$: $\{X_1, \ldots, X_n\} \to \{0,1\}^l$ by:

$$Z_{\vec{z},y}(X_j) \;=\; \begin{cases} \vec{z}(X_j) & \text{if } X_j \neq x_1 \cdots x_i \\ y \oplus x_{i+1}^1 & \text{otherwise.} \end{cases}$$

Let $Y_{\vec{z},y}$ denote the input labeling induced by $Z_{\vec{z},y}$, and observe that it is given by

$$Y_{\vec{z},y}(X_j) \;=\; \begin{cases} \vec{y}(X_j) & \text{if } X_j \notin \textsc{children}(x_1 \cdots x_i) \\ y \oplus x_{i+1}^1 \oplus x_{i+1}^u & \text{if } X_j = x_1 \cdots x_i x_{i+1}^u \text{ for some } 1 \le u \le s. \end{cases}$$

Let $S(\vec{z})$ be the set of all strings y such that $Z_{\vec{z},y}$ is a collision free labeling. We leave to the reader to check that $y \notin S(\vec{z})$ if and only if one of the following two conditions is satisfied:

(1) Either $y \oplus x_{i+1}^1 \in \{\, \vec{z}(X_j) \;:\; 1 \le j \le n \text{ and } X_j \ne x_1 \cdots x_i \,\}$; or

(2) For some $u \in \{1, \ldots, s\}$ it is the case that $y \oplus x_{i+1}^1 \oplus x_{i+1}^u \in \{\, \vec{y}(X_j) \;:\; 1 \le j \le n \text{ and } X_j \notin \textsc{children}(x_1 \cdots x_i) \,\}$.

This implies that $|\{0,1\}^l - S(\vec{z})| \le (n-1) + (n-s)s \le n - 1 + n^2/4 \le 2^l/2$. So $|S(\vec{z})| \ge 2^l - 2^l/2 \ge 2^l/2$. Now observe that any collision free labeling equals $Z_{\vec{z},y}$ for some $\vec{z}, y$ as above. Furthermore by Lemma 3.6 all collision free labelings are equally likely. From this one can prove the desired statement.

Proof of **(2).** The idea is very similar to the above. This time, observe that with $\vec{z}$ given, if we assign an output label $z \in \{0,1\}^l$ to $x_1 \cdots x_i$ then the values of both Z_n and Y_n are fully determined for all nodes $X_1, \ldots, X_n$. We show as before that there is a set $S(\vec{z})$ of these z values for which the determined labeling is collision free, and conclude as before using the equi-probability of collision free labelings. The formal proof follows.

Let $z \in \{0,1\}^l$ be some fixed string. Now define the labeling $Z_{\vec{z},z}$: $\{X_1, \ldots, X_n\} \to \{0,1\}^l$ by:

$$Z_{\vec{z},z}(X_j) \;=\; \begin{cases} \vec{z}(X_j) & \text{if } X_j \ne x_1 \cdots x_i \\ z & \text{otherwise.} \end{cases}$$

Let $Y_{\vec{z},z}$ denote the input labeling induced by $Z_{\vec{z},z}$, and observe that it is given by

$$Y_{\vec{z},z}(X_j) \;=\; \begin{cases} \vec{y}(X_j) & \text{if } X_j \notin \textsc{children}(x_1 \cdots x_i) \\ z \oplus x_{i+1}^u & \text{if } X_j = x_1 \cdots x_i x_{i+1}^u \text{ for some } 1 \le u \le s. \end{cases}$$

Let $S(\vec{z})$ be the set of all strings z such that $Z_{\vec{z},z}$ is a collision free labeling. We leave to the reader to check that $z \notin S(\vec{z})$ if and only if one of the following two conditions is satisfied:

(1) Either $z \in \{\, \vec{z}(X_j) \;:\; 1 \le j \le n \text{ and } X_j \ne x_1 \cdots x_i \,\}$; or

(2) For some $u \in \{1, \ldots, s\}$ it is the case that $z \oplus x_{i+1}^u \in \{\, \vec{y}(X_j) \;:\; 1 \le j \le n \text{ and } X_j \notin \textsc{children}(x_1 \cdots x_i) \,\}$.

This implies that $|\{0,1\}^l - S(\vec{z})| \le (n-1) + (n-s)s \le n - 1 + n^2/4 \le 2^l/2$. So $|S(\vec{z})| \ge 2^l/2$. Now observe that any collision free labeling equals $Z_{\vec{z},z}$ for some $\vec{z}, z$ as above. Furthermore by Lemma 3.6 all collision free labelings are equally likely. From this one can prove the desired statement. ∎

Bounding the probability of collisions. The following lemma fixes the number n of queries that A has made, as well as a particular view $X_1, \ldots, X_n$; $\hat{Z}$ of A. It now makes the assumption that the current labeling Z_n is collision free; think of this fact as being known to A. Given all this, it considers A's adding a new node X_{n+1} to the tree. It says that the labeling is likely to retain its collision freeness; that is, Z_{n+1} is collision free with high probability. The same technical condition on n as in the previous lemma is required.

Note that X_{n+1} is determined by $X_1, \ldots, X_n$; $\hat{Z}$. The value $\mathsf{Z}_{n+1}(X_{n+1})$ has not yet been returned to A, and it makes sense to discuss the distribution of this value given $X_1, \ldots, X_n$; $\hat{Z}$.

Lemma 3.8 Let $1 \le n \le qm - 1$ and suppose $n^2/4 + n - 1 \le 2^l/2$. Let $X_1, \ldots, X_n$ be a query sequence and let $\hat{Z}$ be a labeling of the border nodes of $X_1, \ldots, X_n$. Then

$$\Pr\left[\, \neg\mathsf{ColFree}(\mathsf{Z}_{n+1}) \;\middle|\; \mathsf{View}_n = (X_1, \ldots, X_n \,;\, \hat{Z}) \wedge \mathsf{ColFree}(\mathsf{Z}_n) \,\right] \;\le\; 3n \cdot 2^{-l} \,,$$

where the probability is taken over the choice of f.

Proof: We use the following notation:

$$\Pr_n[\cdot] \;=\; \Pr\left[\,\cdot \mid \mathsf{View}_n = (X_1,\ldots,X_n\,;\,\hat{Z}) \,\wedge\, \mathsf{ColFree}(\mathsf{Z}_n)\,\right] .$$

Case 1. X_{n+1} is at level one.

Let $X_{n+1} = \bar{x}_1$. Note its input label is by definition $\bar{x}_1$. For each $t = 1,\ldots,n$ we claim that

$$\Pr_n[\,\mathsf{Y}_n(X_t) = \bar{x}_1\,] \;\leq\; 2\cdot 2^{-l} . \tag{14}$$

To see why this is true, consider two cases. First, if X_t is at level one then $\Pr_n[\,\mathsf{Y}_n(X_t) = \bar{x}_1\,] = 0$ by definition. On the other hand suppose X_t is at depth at least two. Then $X_t = x_1\cdots x_i x_{i+1}$ is the child of some $x_1\cdots x_i \in \{X_1,\ldots,X_n\}$. Equation 14 now follows by Part 1 of Lemma 3.7.

If $\bar{x}_1 \notin \{\mathsf{Y}_n(X_1),\ldots,\mathsf{Y}_n(X_n)\}$, then even conditioned on View_n, $\mathsf{Z}_{n+1}(X_{n+1})$ will be uniformly distributed over l-bit strings. Given this observation and Equation 14 we can bound the probability of a collision as follows:

$$\begin{aligned}
\Pr_n[\,\neg\mathsf{ColFree}(\mathsf{Z}_{n+1})\,] \;&\leq\; \Pr_n[\,\bar{x}_1 \in \{\mathsf{Y}_n(X_1),\ldots,\mathsf{Y}_n(X_n)\}\,] \,+ \\
&\quad \Pr_n[\,\mathsf{Z}_{n+1}(X_{n+1}) \in \{\mathsf{Z}_n(X_1),\ldots,\mathsf{Z}_n(X_n)\} \mid \bar{x}_1 \notin \{\mathsf{Y}_n(X_1),\ldots,\mathsf{Y}_n(X_n)\}\,] \\
&\leq\; \frac{2n}{2^l} + \frac{n}{2^l} \;\leq\; \frac{3n}{2^l} .
\end{aligned}$$

Case 2. X_{n+1} is not at level one.

Then $X_{n+1} = x_1\cdots x_i x_{i+1}$ is the child of some $x_1\cdots x_i \in \{X_1,\ldots,X_n\}$. Let $S = \{X_1,\ldots,X_n\} - \{x_1\cdots x_i\}$. We first claim that for any $X_t \in \{X_1,\ldots,X_n\}$:

$$\Pr_n[\,\mathsf{Y}_{n+1}(X_{n+1}) = \mathsf{Y}_n(X_t)\,] \;\leq\; 2\cdot 2^{-l} . \tag{15}$$

To see why this is true, consider two cases. First, if X_t is a sibling of X_{n+1} then

$$\Pr_n[\,\mathsf{Y}_{n+1}(X_{n+1}) = \mathsf{Y}_n(X_t)\,] \;=\; 0$$

by definition. On the other hand suppose X_t is not a sibling of X_{n+1}. Then a collision free labeling $\vec{z}$ of S determines $\mathsf{Y}_n(X_t)$. Using this and Part 2 of Lemma 3.7 we have the following: (The sum here is over all collision free labelings $\vec{z}$ of S which are consistent with $\hat{Z}$.)

$$\begin{aligned}
&\Pr_n[\,\mathsf{Y}_{n+1}(X_{n+1}) = \mathsf{Y}_n(X_t)\,] \\
&= \textstyle\sum_{\vec{z}} \Pr_n\left[\,\mathsf{Y}_{n+1}(X_{n+1}) = \mathsf{Y}_n(X_t) \mid \mathsf{Z}_n^S = \vec{z}\,\right]\cdot \Pr_n\left[\,\mathsf{Z}_n^S = \vec{z}\,\right] \\
&= \textstyle\sum_{\vec{z}} \Pr_n\left[\,\mathsf{Z}_n(x_1\cdots x_i) = \mathsf{Y}_n(X_t)\oplus x_{i+1} \mid \mathsf{Z}_n^S = \vec{z}\,\right]\cdot \Pr_n\left[\,\mathsf{Z}_n^S = \vec{z}\,\right] \\
&\leq \frac{2}{2^l}\cdot \textstyle\sum_{\vec{z}} \Pr_n\left[\,\mathsf{Z}_n^S = \vec{z}\,\right] \;\leq\; \frac{2}{2^l} .
\end{aligned}$$

Thus Equation 15 is again established.

Given Equation 15 we can bound the probability of a collision:

$$\Pr_n[\,\neg\mathsf{ColFree}(\mathsf{Z}_{n+1})\,] \;\leq\; \Pr_n[\,\mathsf{Y}_{n+1}(X_{n+1}) \in \{\mathsf{Y}_n(X_1),\ldots,\mathsf{Y}_n(X_n)\}\,] \,+$$

$$\Pr_n [\, \mathsf{Z}_{n+1}(X_{n+1}) \in \{\mathsf{Z}_n(X_1), \ldots, \mathsf{Z}_n(X_n)\} \mid \mathsf{Y}_{n+1}(X_{n+1}) \notin \{\mathsf{Y}_n(X_1), \ldots, \mathsf{Y}_n(X_n)\} \,]$$

$$\leq \frac{2n}{2^l} + \frac{n}{2^l} \leq \frac{3n}{2^l} .$$

This completes the proof of Lemma 3.8. ▮

Concluding the proof. We need to show that

$$\mathbf{Adv}^{\mathrm{dist}}_{\mathsf{CBC}^m\text{-}\mathrm{Rand}^{l\to l},\mathrm{Rand}^{ml\to l}}(A) \quad \leq \quad 1.5 \cdot \frac{q^2m^2}{2^l} .$$

Let $\Pr_1[\cdot]$ denote the probability when A's oracle g is chosen via $f \stackrel{R}{\leftarrow} \mathrm{Rand}^{l\to l}$; $g \leftarrow f^{(m)}$, and let $\mathsf{CF} = \mathsf{ColFree}(\mathsf{Z}_{qm-1})$. Let $\Pr_2[\cdot]$ denote the probability when A's oracle g is chosen via $g \stackrel{R}{\leftarrow} \mathrm{Rand}^{ml\to l}$. We claim that

$$\Pr_1[\, A^g = 1 \mid \neg\mathsf{CF} \,] \quad = \quad \Pr_2[\, A^g = 1 \,] . \tag{16}$$

This is true because as long as the current labeling Z_n which A has is collision-free, the value of a border node returned to A by $g = f^{(m)}$ is a random l-bit string distributed independently of anything else. Thus the distribution on A's view is the same as if A were replied to by a random function g from $\mathrm{Rand}^{ml\to l}$. Now using Equation (16) we have

$$\begin{aligned}
&\mathbf{Adv}^{\mathrm{dist}}_{\mathsf{CBC}^m\text{-}\mathrm{Rand}^{l\to l},\mathrm{Rand}^{ml\to l}}(A) \\
&= \Pr_1[\, A^g = 1 \,] - \Pr_2[\, A^g = 1 \,] \\
&= \Pr_1[\, A^g = 1 \mid \mathsf{CF} \,] \cdot \Pr_1[\, \mathsf{CF} \,] + \Pr_1[\, A^g = 1 \mid \neg\mathsf{CF} \,] \cdot \Pr_1[\, \neg\mathsf{CF} \,] - \Pr_2[\, A^g = 1 \,] \\
&= \Pr_1[\, A^g = 1 \mid \mathsf{CF} \,] \cdot \Pr_1[\, \mathsf{CF} \,] + \Pr_2[\, A^g = 1 \,] \cdot \Pr_1[\, \neg\mathsf{CF} \,] - \Pr_2[\, A^g = 1 \,] \\
&\leq \Pr_1[\, \mathsf{CF} \,] + \Pr_2[\, A^g = 1 \,] \cdot (\Pr_1[\, \neg\mathsf{CF} \,] - 1) \\
&\leq \Pr_1[\, \mathsf{CF} \,] \\
&\leq \textstyle\sum_{n=1}^{qm-2} \Pr_1[\, \neg\mathsf{ColFree}(\mathsf{Z}_{n+1}) \mid \mathsf{ColFree}(\mathsf{Z}_n) \,] \\
&\leq 3(qm-2)(qm-1) \cdot 2^{-l-1} .
\end{aligned}$$

The last inequality follows from Lemma 3.8, which can be applied since $qm \leq 2^{(l+1)/2}$ implies $n^2/4 + n - 1 \leq 2^l/2$ for all $n = 1, \ldots, qm-2$. This concludes the proof.

4 Security of CBC as a MAC

The previous section showed that the CBC-MAC of a PRF is itself a PRF. Recall our original goal was to assess the security of the CBC-MAC as a MAC. In other words, we want to assess the resistance to forgery rather than the indistinguishability with respect to random functions. This is easily done given what we now know. Below we begin by stating the corresponding theorem. Then we go on to ask whether

4.1 Upper bounding the MAC insecurity of CBC

The following theorem bounds the MAC insecurity of $\mathsf{CBC}^m\text{-}F$ (as defined in Section 2.4) in terms of the insecurity of F as a PRF or PRP, thereby saying that if F is a PRF or PRP then its CBC-MAC is a secure MAC.

Theorem 4.1 [CBC Theorem: Security as a MAC] Let $l, m \geq 1$ and $q, t \geq 1$ be integers such that $qm \leq 2^{(l+1)/2}$. Let F: $Keys(F) \times \{0,1\}^l \to \{0,1\}^l$ be a family of functions. Then

$$\begin{aligned}\mathbf{Adv}^{\mathrm{mac}}_{\mathsf{CBC}^m\text{-}F}(q,t) &\leq \mathbf{Adv}^{\mathrm{prf}}_{F}(q',t') + \frac{3q^2m^2+2}{2^{l+1}} \\ &\leq \mathbf{Adv}^{\mathrm{prp}}_{F}(q',t') + \frac{2q^2m^2+1}{2^{l}}\end{aligned}$$

where $q' = mq$ and $t' = t + O(mql)$.

Once again the O-notation conceals a small model-dependent constant.

Proof: Applying first Proposition 2.7 and then Theorem 3.2 we get

$$\begin{aligned}\mathbf{Adv}^{\mathrm{mac}}_{\mathsf{CBC}^m\text{-}F}(q,t) &\leq \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CBC}^m\text{-}F}(q,t+O(ml)) + \frac{1}{2^l} \\ &\leq \mathbf{Adv}^{\mathrm{prf}}_{F}(q',t') + 1.5 \cdot \frac{q^2m^2}{2^l} + \frac{1}{2^l} \ . \end{aligned} \tag{17}$$

Simplifying the right hand side yields the first inequality of the theorem. Now we continue, noting that

$$\begin{aligned}\mathbf{Adv}^{\mathrm{mac}}_{\mathsf{CBC}^m\text{-}F}(q,t) &\leq \mathbf{Adv}^{\mathrm{prf}}_{F}(q',t') + \frac{3q^2m^2+2}{2^{l+1}} \\ &\leq \mathbf{Adv}^{\mathrm{prp}}_{F}(q',t') + \frac{3q^2m^2+2}{2^{l+1}} + \frac{q'(q'-1)}{2^{l+1}} \\ &\leq \mathbf{Adv}^{\mathrm{prp}}_{F}(q',t') + \frac{3q^2m^2+2}{2^{l+1}} + \frac{q^2m^2}{2^{l+1}} \ .\end{aligned}$$

Simplifying the right hand side yields the second inequality of the theorem. ▮

Numerical example (continued). Suppose we use the CBC MAC where the underlying block cipher is the AES algorithm (so $l = 128$). Suppose some scientist finds a practical method which has a 1% chance of forging messages after having asked for the MACs of $q = 2^{30}$ messages, each 16 KBytes long. Then there is some equally practical method—it's running time is essentially that of forging algorithm—which has advantage of at least $0.01 - 7.2 \times 10^{-15} \approx 1\%$ at differentiating AES values on 2^{40} points from as many random distinct points.

Discussion. Our approach to proving the security of the CBC-MAC as a MAC has been to prove something stronger, namely that it is a PRF. This works because any PRF is a MAC (Proposition 2.7). However the converse is not true: not every MAC is a PRF. Indeed, indistinguishability from a random function is a much stronger property than unforgeability. This raises the question of whether better results on the unforgeability of the CBC-MAC could be obtained by directly trying to analyze it as a MAC. In other words, perhaps bounds on $\mathbf{Adv}^{\mathrm{mac}}_{\mathsf{CBC}^m\text{-}F}(q,t)$ much better than those of Theorem 4.1 could be obtained via an analytical approach different from the one we have taken.

There is room for improvement via alternative approaches. In the next subsection we show that the quadratic dependence on the number of queries q in the insecurity function of the CBC-MAC is necessary. Thus at best one might hope to reduce the dependency on the number m of blocks in the messages.

4.2 Birthday attack on the CBC MAC

The basic idea behind the attack, due to Preneel and Van Oorschott [19] and (independently) to Krawczyk, is that internal collisions can be exploited for forgery. Here we use this idea to present an attack on the CBC-MAC in the case that the underlying family is a family of permutations. (We focus on this case because in practice the CBC-MAC is usually based on a block cipher.)

The attacks presented in [19] are analyzed assuming the underlying functions are random, meaning the family to which the CBC-MAC transform is applied is $\mathrm{Rand}^{l\to l}$ or Perm^l. Here we do not make such an assumption. The attack we present works for any family of permutations. The randomness in our attack (which is the source of birthday collisions) comes from coin tosses of the forger only. This makes the attack more general.

Proposition 4.2 Let l, m, q be integers such that $1 \le q \le 2^{(l+1)/2}$ and $m \ge 2$. Let F: $\mathit{Keys}(F) \times \{0,1\}^l \to \{0,1\}^l$ be a family of permutations. Then there is a forger A making $q+1$ oracle queries, running for time $O(lmq\ \log q)$ and achieving

$$\mathbf{Adv}^{\mathrm{mac}}_{\mathsf{CBC}^m\text{-}F}(A) \quad \ge \quad 0.316 \cdot \frac{q(q-1)}{2^l}\ .$$

As a consequence for $q \ge 2$

$$\mathbf{Adv}^{\mathrm{mac}}_{\mathsf{CBC}^m\text{-}F}(q,t) \quad \ge \quad 0.316 \cdot \frac{(q-1)(q-2)}{2^l}\ .$$

The time assessment here puts the cost of an oracle call at one unit.

Comparing the above to Theorem 4.1 we see that our upper bound is tight to within a factor of the square of the number of message blocks.

We now proceed to the proof. We begin with a couple of lemmas. The first lemma considers a slight variant of the usual birthday problem and shows that the "collision probability" is still the same as that of the usual birthday problem.

Lemma 4.3 Let l, q be integers such that $1 \le q \le 2^{(l+1)/2}$. Fix $b_1, \ldots, b_q \in \{0,1\}^l$. Then

$$\Pr\left[\, r_1, \ldots, r_q \stackrel{R}{\leftarrow} \{0,1\}^l \ :\ \exists\, i,j \text{ such that } i \ne j \text{ and } b_i \oplus r_i = b_j \oplus r_j \,\right] \quad \ge \quad 0.316 \cdot \frac{q(q-1)}{2^l}\ .$$

Proof: This is just like throwing q balls into $N = 2^l$ bins and lower bounding the collision probability, except that things are "shifted" a bit: the bin assigned to the i-th ball is $r_i \oplus b_i$ rather than r_i as we would usually imagine. But with b_i fixed, if r_i is uniformly distributed, so is $r_i \oplus b_i$. So the probabilities are the same as in the standard birthday problem of Appendix A. ∎

The first part of the following lemma states an obvious property of the CBC-MAC transform. The item of real interest is the second part of the lemma, which says that in the case where the underlying function is a permutation, the CBC-MAC transform has the property that output collisions occur if and only if input collisions occur. This is crucial to the attack we will present later.

Lemma 4.4 Let $l, m \geq 2$ be integers and $f \colon \{0,1\}^l \to \{0,1\}^l$ a function. Suppose $\alpha_1 \cdots \alpha_m$ and $\beta_1 \cdots \beta_m$ in $\{0,1\}^{ml}$ are such that $\alpha_k = \beta_k$ for $k = 3, \ldots, m$. Then

$$f(\alpha_1) \oplus \alpha_2 = f(\beta_1) \oplus \beta_2 \quad \Rightarrow \quad f^{(m)}(\alpha_1 \cdots \alpha_m) = f^{(m)}(\beta_1 \cdots \beta_m) .$$

If f is a permutation then, in addition, the converse is true:

$$f^{(m)}(\alpha_1 \cdots \alpha_m) = f^{(m)}(\beta_1 \cdots \beta_m) \quad \Rightarrow \quad f(\alpha_1) \oplus \alpha_2 = f(\beta_1) \oplus \beta_2 .$$

Proof: The first part follows from the definition of $f^{(m)}$. For the second part let f^{-1} denote the inverse of the permutation f. The CBC-MAC computation is easily unraveled using f^{-1}. Thus the procedure

$$y_m \leftarrow f^{(m)}(\alpha_1 \cdots \alpha_m) \,; \quad \text{For } k = m \text{ downto } 3 \text{ do } y_{k-1} \leftarrow f^{-1}(y_k) \oplus \alpha_k \text{ End For } \,; \text{ Return } f^{-1}(y_2)$$

returns $f(\alpha_1) \oplus \alpha_2$, while the procedure

$$y_m \leftarrow f^{(m)}(\beta_1 \cdots \beta_m) \,; \quad \text{For } k = m \text{ downto } 3 \text{ do } y_{k-1} \leftarrow f^{-1}(y_k) \oplus \beta_k \text{ End For } \,; \text{ Return } f^{-1}(y_2)$$

returns $f(\beta_1) \oplus \beta_2$. But the procedures have the same value of y_m by assumption and we know that $\alpha_k = \beta_k$ for $k = 3, \ldots, m$, so the procedures return the same thing. ▮

Proof of Proposition 4.2: Before presenting the forger let us discuss the idea.

The forger A has an oracle $g = f^{(m)}$ where f is an instance of F. The strategy of the forger is to make q queries all of which agree in the last $m - 2$ blocks. The first blocks of these queries are all distinct but fixed. The second blocks, however, are random and independent across the queries. Denoting the first block of query n by a_n and the second block as r_n, the forger hopes to have $i \neq j$ such that $f(a_i) \oplus r_i = f(a_j) \oplus r_j$. The probability of this happening is lower bounded by Lemma 4.3, but simply knowing the event happens with some probability is not enough; the forger needs to detect its happening. Lemma 4.4 enables us to say that this internal collision happens iff the output MAC values for these queries are equal. (This is true because f is a permutation.) We then observe that if the second blocks of the two colliding queries are modified by the xor to both of some value a, the resulting queries still collide. The forger can thus forge by modifying the second blocks in this way, obtaining the MAC of one of the modified queries using the second, and outputting it as the MAC of the second modified query.

The forger is presented in detail below. It makes use of a subroutine FindCol that given a sequence $\sigma_1, \ldots, \sigma_q$ of values returns a pair (i, j) such that $\sigma_i = \sigma_j$ if such a pair exists, and otherwise returns $(0, 0)$.

Forger A^g

Let $a_1, \ldots, a_q$ be distinct l-bit strings
For $i = 1, \ldots, q$ do $r_i \stackrel{R}{\leftarrow} \{0,1\}^l$
For $i = 1, \ldots, q$ do
  $x_{i,1} \leftarrow a_i$; $x_{i,2} \leftarrow r_i$
  For $k = 3, \ldots, m$ do $x_{i,k} \leftarrow 0^l$
  $X_i \leftarrow x_{i,1} \ldots x_{i,m}$
  $\sigma_i \leftarrow g(X_i)$
End For
$(i,j) \leftarrow \text{FindCol}(\sigma_1, \ldots, \sigma_q)$
If $(i,j) = (0,0)$ then abort
Else
  Let a be any l-bit string different from 0^l
  $x'_{i,2} \leftarrow x_{i,2} \oplus a$; $x'_{j,2} \leftarrow x_{j,2} \oplus a$
  $X'_i \leftarrow x_{i,1} x'_{i,2} x_{i,3} \cdots x_{i,m}$; $X'_j \leftarrow x_{j,1} x'_{j,2} x_{j,3} \cdots x_{j,m}$
  $\sigma'_i \leftarrow g(X'_i)$
  Return (X'_j, σ'_i)
End If

To estimate the probability of success, suppose $g = f^{(m)}$ where f is an instance of F. Let (i,j) be the pair of values returned by the FindCol subroutine. Assume $(i,j) \neq (0,0)$. Then we know that

$$f^{(m)}(x_{i,1} \cdots x_{i,m}) = f^{(m)}(x_{j,1} \cdots x_{j,m}) .$$

By assumption f is a permutation and by design $x_{i,k} = x_{j,k}$ for $k = 3, \ldots, m$. The second part of Lemma 4.4 then implies that $f(a_i) \oplus r_i = f(a_j) \oplus r_j$. Adding a to both sides we get $f(a_i) \oplus (r_i \oplus a) = f(a_j) \oplus (r_j \oplus a)$. In other words, $f(a_i) \oplus x'_{i,2} = f(a_j) \oplus x'_{j,2}$. The first part of Lemma 4.4 then implies that $f^{(m)}(X'_i) = f^{(m)}(X'_j)$. Thus σ'_i is a correct MAC of X'_j. Furthermore we claim that X'_j is new, meaning was not queried of the g oracle. Since $a_1, \ldots, a_q$ are distinct, the only thing we have to worry about is that $X'_j = X_j$, but this is ruled out because $a \neq 0^l$.

We have just argued that if the FindCol subroutine returns $(i,j) \neq (0,0)$ then the forger is successful, so the success probability is the probability that $(i,j) \neq (0,0)$. This happens whenever here is a collision amongst the q values $\sigma_1, \ldots, \sigma_q$. Lemma 4.4 tells us however that there is a collision in these values if and only if there is a collision amongst the q values $f(a_1) \oplus r_1, \ldots, f(a_q) \oplus r_q$. The probability is over the random choices of $r_1, \ldots, r_q$. By Lemma 4.3 the probability of the latter is lower bounded by the quantity claimed in the Proposition. We conclude the theorem by noting that, with a simple implementation of FindCol (say using a balanced binary search tree scheme) the running time is as claimed. ∎

5 Length Variability

For simplicity, let us assume throughout this section that strings to be authenticated have length which is a multiple of l bits. This restriction is easy to dispense with by using simple and well-known padding methods: for example, always append a "1" and then append the minimal number of 0's to make the string a multiple of l bits.

The CBC MAC doesn't handle variable-length inputs. The CBC MAC does not directly give a method to authenticate messages of variable input lengths. In fact, it is easy to "break" the CBC MAC construction if the length of strings is allowed to vary (this fact is well-known). As an example, if an adversary requests $f_a^{(1)}$ of b, obtaining t_b, and then requests $f_a^{(1)}(t_b)$, obtaining t_{t_b}, it can then compute the MAC $f_a^{(2)}(b\|0) = t_{t_b}$ for $b\|0$—a string for which it has not asked the MA.C

Appending the length doesn't work. One possible attempt to authenticate messages of varying lengths is to append to each string $x = x_1 \cdots x_m$ the number m, properly encoded as the final l-bit block, and then CBC MAC the resulting string $m+1$ blocks. (Of course this imposes a restriction that $m < 2^l$, not likely to be a serious concern.) We define $f_a^*(x_1 \cdots x_m) = f_a^{(m+1)}(x_1 \cdots x_m\ m)$.

We show that f^* is not a secure MAC. Take arbitrary l-bit words b, b' and c, where $b \neq b'$. It is easy to check that given

(1) $t_b = f^*(b)$,

(2) $t_{b'} = f^*(b')$, and

(3) $t_{b1c} = f^*(b\|1\|c)$

the adversary has in hand $f^*(b'\|1\|t_b \oplus t_{b'} \oplus c)$ —the authentication tag of a string she has not asked about before—since this is precisely t_{b1c}.

Better methods. Despite the failure of the above method there are many suitable ways to obtain a PRF that is good on variable input lengths. We mention three. In each, let F be a finite function family from and to l-bit strings. Let $x = x_1 \cdots x_m$ be the message to which we will apply f_a:

(1) *Input-length key separation.* Set $f_a^*(x) = f_{a_m}^{(m)}(x)$, where $a_m = f_a(m)$.

(2) *Length-prepending.* Set $f_a^*(x) = f_a^{(m+1)}(m\|x)$.

(3) *Encrypt last block.* Set $f_{a_1 a_2}^*(x) = f_{a_2}(f_{a_1}^{(m)}(x))$.

The last method appears in an informational Annex of [13], and has now been analyzed by Petrank and Rackoff [18]. It is the most attractive method of the bunch, since the length of x is not needed until the end of the computation, facilitating on-line MAC computation. One additional method was mentioned in the proceedings version of this paper (the "two-step MAC," [4, p. 352]), but Petrank and Rackoff have pointed out that this method does not work [18].

Acknowledgments

We thank Uri Feige and Moni Naor for their assistance in the proof of Theorem 3.1, and also for comments on the paper.

References

[1] J. An and M. Bellare, Constructing VIL-MACs from FIL-MACs: Message authentication under weakened assumptions, *Advances in Cryptology – CRYPTO '99*, Springer-Verlag, Lecture Notes in Computer Science, Vol. ??, M. Wiener, Ed., pp. ??–??, 1999.

[2] ANSI X9.9, American National Standard for Financial Institution Message Authentication (Wholesale), American Bankers Association, 1981. Revised 1986.

[3] M. Bellare, A. Desai, E. Jokipii and P. Rogaway, A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation, *Proceedings of the* 38th *Symposium on Foundations of Computer Science*, IEEE, 1997.

[4] M. Bellare, J. Kilian and P. Rogaway, The security of cipher block chaining, *Advances in Cryptology – CRYPTO '94,* Lecture Notes in Computer Science Vol. 839, Springer-Verlag, Y. Desmedt, Ed., pp. 340–358, 1994.

[5] M. Bellare, R. Canetti, and H. Krawczyk, Keying hash functions for message authentication", *Advances in Cryptology – CRYPTO '96*, Springer-Verlag, Lecture Notes in Computer Science, Vol. 1109, N. Koblitz, Ed., pp. 1–15, 1996.

[6] M. Bellare, R. Guérin and P. Rogaway, XOR MACs: New methods for message authentication using finite pseudorandom functions, *Advances in Cryptology – CRYPTO '95,* Lecture Notes in Computer Science, Vol. 963, Springer-Verlag, D. Coppersmith, Ed., pp. 15–28, 1995.

[7] M. Bellare and P. Rogaway, Entity authentication and key distribution, *Advances in Cryptology – CRYPTO '93,* Lecture Notes in Computer Science, Vol. 773, Springer-Verlag, D. Stinson, Ed., pp. 232–249, 1993.

[8] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva and M. Yung, Systematic design of two-party authentication protocols, *Advances in Cryptology – Crypto* 91 *Proceedings*, Lecture Notes in Computer Science Vol. 576, J. Feigenbaum ed., Springer-Verlag, 1991.

[9] J. Black, S. Halevi, H. Krawczyk, T. Krovetz and P. Rogaway, UMAC: Fast and secure message authentication, *Advances in Cryptology – CRYPTO '99* Lecture Notes in Computer Science, Vol. ??, Springer-Verlag, M. Wiener., Ed., pp ??–??, 1999.

[10] O. Goldreich, S. Goldwasser and S. Micali, How to construct random functions, *Journal of the ACM,* Vol. 33, No. 4, 210–217, (1986).

[11] O. Goldreich, S. Goldwasser and S. Micali, On the cryptographic applications of random functions, *Advances in Cryptology – Crypto* 84 *Proceedings*, Lecture Notes in Computer Science Vol. 196, R. Blakely ed., Springer-Verlag, 1984.

[12] S. Goldwasser, S. Micali and R. Rivest, A digital signature scheme secure against adaptive chosen-message attacks, *SIAM Journal of Computing,* 17(2):281–308, April 1988.

[13] ISO/IEC 9797, Data cryptographic techniques – Data integrity mechanism using a cryptographic check function employing a block cipher algorithm, 1989.

[14] L. Knudsen, A chosen text attack on CBC-MAC, *Electronics Letters* 33(1), 1997, pp. 48–49.

[15] M. Luby and C. Rackoff, How to construct pseudorandom permutations from pseudorandom functions, *SIAM J. Computation,* Vol. 17, No. 2, April 1988.

[16] M. Luby and C. Rackoff, A study of password security, *Advances in Cryptology – Crypto* 87 *Proceedings*, Lecture Notes in Computer Science Vol. 293, C. Pomerance ed., Springer-Verlag, 1987.

[17] K. Ohta and M. Matsui, Differential attack on message authentication codes, *Advances in Cryptology – Crypto* 93 *Proceedings*, Lecture Notes in Computer Science Vol. 773, D. Stinson ed., Springer-Verlag, 1993.

[18] E. Petrank and C Rackoff, CBC MAC for real-time data sources, manuscript, 1997.

[19] B. Preneel and P. van Oorschot, MDx-MAC and building fast MACs from hash functions, *Advances in Cryptology – CRYPTO '95*, Lecture Notes in Computer Science Vol. 963, Springer-Verlag, D. Coppersmith, Ed., pp. 1–14, 1995.

[20] S. Stubblebine and V. Gligor, On message integrity in cryptographic protocols, *Proceedings of the 1992 IEEE Computer Society Symposium on Research in Security and Privacy.* May 1992.

[21] M. Wegman and L. Carter, New hash functions and their use in authentication and set equality, *J. of Computer and System Sciences* 22, 265–279 (1981).

A Birthday Bounds

Many of our estimates require precise bounds on the birthday probabilities which for completeness we derive here.

The setting is that we have q balls. View them as numbered, $1, \ldots, q$. We also have N bins, where $N \geq q$. We throw the balls at random into the bins, one by one, beginning with ball 1. At random means that each ball is equally likely to land in any of the N bins, and the probabilities for all the balls are independent. A collision is said to occur if some bin ends up containing at least two balls. We are interested in $C(N, q)$, the probability of a collision.

The birthday phenomenon takes its name from the case when $N = 365$, whence we are asking what is the chance that, in a group of q people, there are two people with the same birthday, assuming birthdays are randomly and independently distributed over the 365 days of the year. It turns out that when q hits $\sqrt{365} \approx 19.1$ the chance of a collision is already quite high; for example at $q = 20$ the chance of a collision is at least 0.328. The following gives upper and lower bounds on this probability.

Proposition A.1 Let $C(N, q)$ denote the probability of at least one collision when we throw $q \geq 1$ balls at random into $N \geq q$ buckets. Then

$$C(N,q) \quad \leq \quad \frac{q(q-1)}{2N} \; .$$

Also

$$C(N,q) \quad \geq \quad 1 - e^{-q(q-1)/2N} \; ,$$

and for $1 \leq q \leq \sqrt{2N}$

$$C(N,q) \quad \geq \quad 0.316 \cdot \frac{q(q-1)}{N} \; .$$

Proof of Proposition A.1: Let C_i be the event that the i-th ball collides with one of the previous ones. Then $\Pr[C_i]$ is at most $(i-1)/N$, since when the i-th ball is thrown in, there are at most $i-1$ different occupied bins and the i-th ball is equally likely to land in any of them. Now

$$\begin{aligned} C(N,q) &= \Pr[C_1 \vee C_2 \vee \cdots \vee C_q] \\ &\leq \Pr[C_1] + \Pr[C_2] + \cdots + \Pr[C_q] \\ &\leq \frac{0}{N} + \frac{1}{N} + \cdots + \frac{q-1}{N} \\ &= \frac{q(q-1)}{2N} . \end{aligned}$$

This proves the upper bound. For the lower bound we let D_i be the event that there is no collision after having thrown in the i-th ball. If there is no collision after throwing in i balls then they must all be occupying different slots, so the probability of no collision upon throwing in the $(i+1)$-st ball is exactly $(N-i)/N$. That is,

$$\Pr[D_{i+1} \mid D_i] = \frac{N-i}{N} = 1 - \frac{i}{N} .$$

Also note $\Pr[D_1] = 1$. The probability of no collision at the end of the game can now be computed via

$$\begin{aligned} 1 - C(N,q) &= \Pr[D_q] \\ &= \Pr[D_q \mid D_{q-1}] \cdot \Pr[D_{q-1}] \\ &\vdots \quad \vdots \\ &= \prod_{i=1}^{q-1} \Pr[D_{i+1} \mid D_i] \\ &= \prod_{i=1}^{q-1} \left(1 - \frac{i}{N}\right) . \end{aligned}$$

Note that $i/N \leq 1$. So we can use the inequality $1 - x \leq e^{-x}$ for each term of the above expression. This means the above is not more than

$$\prod_{i=1}^{q-1} e^{-i/N} = e^{-1/N-2/N-\cdots-(q-1)/N} = e^{-q(q-1)/2N} .$$

Putting all this together we get

$$C(N,q) \geq 1 - e^{-q(q-1)/2N} ,$$

which is the second inequality in Proposition A.1. Finally, to get the last inequality in the theorem statement, we we know $q(q-1)/2N \leq 1$ because $q \leq \sqrt{2N}$, so we can use the inequality $1 - e^{-x} \geq (1 - e^{-1})x$ to get

$$C(N,q) \geq \left(1 - \frac{1}{e}\right) \cdot \frac{q(q-1)}{2N} .$$

Noting that $(1 - 1/e)/2 > 0.316$ completes the proof. ∎