A preliminary version of this paper appeared in *Advances in Cryptology – Crypto 95 Proceedings*, Lecture Notes in Computer Science Vol. 963, D. Coppersmith ed., Springer-Verlag, 1995.

XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions

Mihir Bellare* Roch Guérin† Phillip Rogaway‡

October 1995

Abstract

We describe a new approach for authenticating messages. Our "XOR MACs" have several nice features, including parallelizability, incrementality, and provable security.

Our method uses any finite pseudorandom function (PRF). The finite PRF can be "instantiated" via DES (yielding an alternative to the CBC MAC), via the compression function of MD5 (yielding an alternative to various "keyed MD5" constructions), or in a variety of other ways.

The proven security is quantitative, expressing the adversary's inability to forge in terms of her (presumed) inability to break the underlying finite PRF. This is backed by attacks showing the analysis is tight. Our proofs exploit linear algebraic techniques, and relate the security of a given XOR scheme to the probability that a certain associated matrix is of full rank.

Our analysis shows that XOR schemes are actually more secure than the CBC MAC, in a sense that we can make precise.

*Department of Computer Science & Engineering, Mail Code 0114, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093. E-mail: `mihir@cs.ucsd.edu`

†Advanced Networking Laboratory, IBM T.J. Watson Research Center, PO Box 704, Yorktown Heights, New York 10598, USA. e-mail: `guerin@watson.ibm.com`.

‡ Dept. of Computer Science, Eng. II Bldg., University of California, Davis, CA 95616, USA. e-mail: `rogaway@cs.ucdavis.edu`.

1 Introduction

A message authentication scheme enables two parties sharing a key a to authenticate their transmissions. This is one of the most widely used cryptographic primitives, and it may become even more so: as security concerns grow, it is reasonable to anticipate that virtually every transmitted message (or packet) will use cryptographic means to ensure authenticity. (For example, the ubiquitous use of message authentication is already being contemplated for the next generation of Internet Protocols.)

Message authentication is usually accomplished by including with each transmitted message M a short string, called its "message authentication code" (MAC) or "signature," computed as a function of M and the shared key a. The most prevalent MAC is the "cipher block chaining message authentication code" (CBC MAC) specified in the International Standard ISO 9797 [ISO] and the U.S. Standard ANSI X9.9 [X9.9]. In recent years another type of MAC has started to become prevalent: these are constructed by somehow "keying" a cryptographic hash, as in $\mathsf{MAC}_a(x) = \mathsf{MD5}(x.a)$ (see, for example, [Ts]).

The goal of the present work is to provide new methods which have certain efficiency and security advantages. We call our methods "XOR schemes." They are simple to describe and implement. They use as their underlying primitive any finite pseudorandom function (PRF). In particular, a finite PRF can be defined from a block cipher (e.g. DES) or from the compression function of a cryptographic hash (e.g., MD5) yielding concrete alternatives to the above mentioned MACs.

What is an XOR MAC? At the highest level, the computation of an XOR MAC consists of three steps: (1) encode the message M as a collection of blocks (each block will depend on a small number of bits from the message); (2) apply the finite PRF to each of the blocks, thus creating a collection of PRF images (the MAC key a is the index for all of these PRF computations); and (3) XOR the set of PRF images together, building the MAC out of the result. Different ways of implement the encoding step (and different choices of the finite PRF) yield different XOR MACs. (Obviously not all encodings will result in secure MACs. We specify several simple ones which do, and also specify general conditions to determine which encodings work.)

This paper specifies, for every finite PRF family F and every value of a block size b, two XOR MACs—a stateless (and probabilistic) one called $\mathsf{XMACR}_{F,b}$, and a stateful (and deterministic) one called $\mathsf{XMACC}_{F,b}$. (In a stateful MAC the signer maintains information, in our case a counter, which he updates each time a message is signed.) The schemes are described concretely in Section 2, as are their main efficiency advantages, namely parallelizability and incrementality.

Security of our schemes. Our XOR schemes are *proven* secure– we show that if the F is a "secure" finite PRF family then the MAC schemes based on it are also "secure." In formalizing this, security of a finite PRF family means indistinguishability from a family of random functions in the sense of [GGM], while security of a MAC means it resists chosen message attack. To make these results meaningful for practice, the security in both cases is made quantitative: we measure the success probabilities as a function of the resources (time and chosen message queries) available to the adversary, and specify exact reductions, enabling the protocol designer to compute, given some assumed security on the finite PRF, how many queries an XOR MAC based on it will withstand. This type of security analysis for a MAC, starting from a finite PRF, begins with [BKR].

Our XOR schemes are so simple that it is tempting to think one can easily find attacks. This is why we stress the importance of the proofs of security which show that no attacks short of breaking the underlying PRF will succeed.

An advantage of quantified security is that it allows one to compare the securities of different MACs based on the same finite PRF family. (Note that a concrete finite PRF family F, eg. a block cipher like DES, may possess strengths which are not reflected in the model of F being a finite PRF family, and these other strengths are potentially relevant in determining the strength of a MAC based on the block cipher. In making security determinations and comparisons we are treating the underlying primitives, eg. DES, as being known to only possess the properties which have been formally modeled, here the property of being a finite PRF.) We will see that our counter based MAC is "more secure" than our randomized one, and that both are "more secure" than the CBC MAC. In particular, the success probability of the adversary in the XOR schemes is independent of the lengths of the messages in her chosen message attack (as long as they stay below a certain specified but very large length) while the attacks of [Kr, PV] show that the success probability of the adversary in the CBC scheme grows as a linear function of the message length. See Section 7.

We also describe the best attacks we know on the XOR schemes. They use birthday attacks (collisions) and indicate that the analysis from our proofs is tight.

2 The Schemes — Concretely

We begin by presenting concrete instantiations of our two main schemes using DES. (But we stress this is just an example. Other instantiations are possible, using other block ciphers, or even methods such as MD5, as discussed later.) We let $l = 64$ and $L = 48$. For any 56-bit key a and l-bit plaintext x we let $F_a(x)$ be the first L bits of $\mathrm{DES}_a(x)$. (We stress that F_a outputs only 48 bits, and not the full 64-bit DES output. We have truncated the output because DES is a pseudorandom *permutation*, while what we want is a pseudorandom function.) Sender and receiver share a 56-bit DES key a which specifies F_a.

Message formatting and notation. We assume the length $|M|$ of M is a multiple of 32 bits, which can easily be achieved by a suitable padding. (For example, append a one and then append enough zeros to bring the length to a multiple of 32 bits.) The message is then viewed as a sequence of 32-bit blocks, $M = M[1] \dots M[n]$ with $|M[i]| = 32$ for $i = 1, \dots, n$. We assume that the number n of blocks is less than 2^{31} —equivalently $|M| \le 32 * 2^{31} = 2^{36}$ bits— which would not normally be a significant restriction in practice.

Let $\langle i \rangle$ denote the binary representation of block index $i \in \{1, \dots, n\}$ as a string of exactly 31 bits. (This is why we assumed the bound on n.) Let $\alpha \,.\, \beta$ denote the concatenation of strings α and β. We give two schemes:

Scheme XMACR. The first scheme is called the randomized XOR scheme, XMACR. To authenticate the message $M = M[1] \dots M[n]$ do the following:

- Pick at random a 63-bit string r, hereafter called the *seed*
- Set $z = F_a(0 \,.\, r) \oplus F_a(1 \,.\, \langle 1 \rangle \,.\, M[1]) \oplus F_a(1 \,.\, \langle 2 \rangle \,.\, M[2]) \oplus \cdots \oplus F_a(1 \,.\, \langle n \rangle \,.\, M[n])$
- Set the MAC of M to the pair $\mu = (r, z)$.

Thus the sender will transmit (M, μ). The receiver, receiving (M', μ'), where $\mu' = (r', z')$, computes $z = F_a(0 \,.\, r') \oplus F_a(1 \,.\, \langle 1 \rangle \,.\, M'[1]) \oplus F_a(1 \,.\, \langle 2 \rangle \,.\, M'[2]) \oplus \cdots \oplus F_a(1 \,.\, \langle n \rangle \,.\, M'[n])$. The receiver accepts M' if and only if $z = z'$.

We stress that new coins are flipped to determine the seed each time the sender wants to authenticate a message, and also that the seed is included in the signature.

Scheme XMACC. The second scheme is called the counter-based XOR scheme. Here it is required that the sender maintain a 63-bit counter C which is initially 0 and is incremented for each message. (Thus at most 2^{63} messages can be signed.) To authenticate message $M = M[1] \dots M[n]$ do the following:

- Increment the counter C by 1
- Set $z = F_a(0 \,.\, C) \oplus F_a(1 \,.\, \langle 1 \rangle \,.\, M[1]) \oplus F_a(1 \,.\, \langle 2 \rangle \,.\, M[2]) \oplus \cdots \oplus F_a(1 \,.\, \langle n \rangle \,.\, M[n])$
- Set the MAC of M to the pair $\mu = (C, z)$.

Thus the sender will transmit (M, μ). The receiver, receiving (M', μ') where $\mu' = (C', z')$, computes $F_a(0 \,.\, C') \oplus F_a(1 \,.\, \langle 1 \rangle \,.\, M'[1]) \oplus F_a(1 \,.\, \langle 2 \rangle \,.\, M'[2]) \oplus \cdots \oplus F_a(1 \,.\, \langle n \rangle \,.\, M'[n])$ and accepts iff this value equals z'. Note the counter is included in the signature. Also the receiver maintains no state.

Stateful schemes are not necessarily "worse" than stateless ones; programmatically, a "static" variable is easy, but a good approximation to randomness is hard. We now discuss properties of XMACR and XMACC.

PARALLELIZABILITY. The DES computations on different blocks can be made in *parallel.* In general, the throughput of an XOR MAC can be doubled by doubling the amount (and not speed) of the underlying hardware. An environment where this is crucial is message authentication over high speed networks (where packets will flow over optical links at rates of 1–10 GBit/second). In that setting one cannot realistically use the CBC MAC because of its sequential nature; an XOR scheme is a more appropriate choice. Note that even in the software setting parallelizability can be relevant: with an adequate degree of parallelism, multiple machine pipelines can all be kept busy doing useful work.

INCREMENTALITY. An XOR MAC is *incremental* [BGG1] with respect to block replacement. Suppose $M[i]$ is modified to a new 32-bit value m. Then, for a long message M, one can update the MAC much quicker than it would take to re-compute it. Let's illustrate for XMACR. Let $\mu = (r, z)$ be a MAC of M and let M' denote M with block i replaced by m. To compute a MAC for M', pick r' at random and let $z' = z \oplus F_a(0 \,.\, r) \oplus F_a(0 \,.\, r') \oplus F_a(1 \,.\, \langle i \rangle \,.\, M[i]) \oplus F_a(1 \,.\, \langle i \rangle \,.\, m)$. Then (r', z') is a MAC for M'. Extensions of this scheme to support insertion and deletion of blocks (not just replacement) appear in [BGG2].

OUT-OF-ORDER VERIFICATION. Tag verification can proceed even if message blocks arrive out of order. Here it is only necessary that the each block be accompanied by its index. With other mechanisms MAC verification cannot proceed before the first block has been received, for example. Out-of-order MAC verification is useful since networks always have some degree of packet loss and re-transmission.

DES COMPUTATIONS. The number of DES computations is twice that of the CBC MAC. (The overhead can be reduced as discussed in Section 4 by increasing the block size, currently set to 32, at the cost of reducing the maximum allowable message length.) So, in software, the above schemes are slower than the CBC MAC. But an XOR MAC based on DES is interesting for hardware efficiency, particularly for high-speed networks, or in settings where the incrementality compensates for the slower from-scratch MACing time. For a software-efficient XOR MAC use the MD5-based instantiation discussed later.

MD5-BASED INSTANTIATION. A software-efficient XOR MAC would start not with DES but with a software-efficient PRF. For example, from the compression function of a cryptographic hash function, say $\mathsf{md} : \{0,1\}^{640} \to \{0,1\}^{128}$, one can define a finite PRF, say $F_a(x)$ equals the first 64 bits of $\mathsf{md}(x \,.\, a)$, where $|x| = 560$ and $|a| = 80$. Using 48-bits for the block index, we would get a MAC which uses one application of md for every 512 bits of message. This is as efficient as proposals like $\mathsf{MD5}(x \,.\, a)$ or $\mathsf{MD5}(a \,.\, x \,.\, a)$ which are currently being considered for the Internet, and has the advantages of parallelizability and incrementality.

SECURITY. Observe that including the block indices in the argument to F_a is necessary— if they are omitted, permuting the message blocks would leave the MAC unchanged. One can also see that the block containing the random string r (resp. counter) of XMACR (resp. XMACC) cannot be omitted. In other words, the scheme in which the MAC is set to $F_a(1 \,.\, \langle 1\rangle \,.\, M[1]) \oplus F_a(1 \,.\, \langle 2\rangle \,.\, M[2]) \oplus \cdots \oplus F_a(1 \,.\, \langle n\rangle \,.\, M[n])$ is easily broken—e.g., set $M_1 = A \,.\, B$, $M_2 = A' \,.\, B$, $M_3 = A \,.\, B'$ and $M_4 = A' \,.\, B'$, and note that the MACs of M_1, M_2, M_3 sum to give the MAC of M_4.

The idea behind the nonces is to prevent the attacker from forming new MACs via linear combinations of old ones. This is in fact the only attack short of breaking the PRF. This is not obvious, of course; indeed it is far from clear why XMACR and XMACC should be secure. That is why we have our proofs.

3 Definitions

Modeling block ciphers as finite pseudorandom functions begins with [BKR]. The underlying notion is the pseudorandom function notion of [GGM], appropriately tailored to take into account the fact that block ciphers have fixed input and output lengths and can't be treated asymptotically, and builds on a suggestion of [LuRa] that DES be viewed as a "pseudorandom in practice" function family.

First some notation. Denote by $x^{(i)}$ the i-th bit of a string x and by $|x|$ its length. If $i \in \{1, \ldots, 2^n\}$ is an integer then we denote by $\langle i\rangle_n$ the natural binary encoding of i as an n-bit string. (Thus the $\langle\cdot\rangle$ of Section 2 is $\langle\cdot\rangle_{31}$ in our current notation.) If S is a set (resp. probability space) then $x \stackrel{R}{\leftarrow} S$ denotes the operation of selecting an element uniformly at random from S (resp. at random according to the distribution specified by S).

3.1 Function families

A *function family* is a set of functions, and an associated set of strings called keys. Each key names a function in the family according to some fixed convention, and the function corresponding to key a is denoted F_a. (Note that two keys can name the same function.) To pick a function f at random from a family F means to pick a key a uniformly at random and let $f = F_a$; we write $f \stackrel{R}{\leftarrow} F$ for this operation. For example DES is a function family where the set of keys is the set of all 56-bit strings.

A family F has input length l and output length L if each $f \in F$ maps $\{0,1\}^l$ to $\{0,1\}^L$. (Eg. $l = L = 64$ for DES.) It has key length κ if the associated set of keys is the set of all strings of length κ. The family of random functions with input length l and output length L is the family R of all functions mapping $\{0,1\}^l$ to $\{0,1\}^L$. The key of a function f in this family is the string which describes its truth table. Note this is a very large family, consisting of 2^{L2^l} functions.

A finite function family F is "pseudorandom" if the input-output behavior of F_a "looks random" to someone who doesn't know the key a. This is formalized via the notion of statistical tests [GGM]. Formally, such a test is an oracle algorithm A. Let F, G be finite function families. The advantage of A in distinguishing F from G is defined by

$$\mathsf{Adv}_A(F,G) \;=\; \Pr_{g \stackrel{R}{\leftarrow} F}[\, A^g = 1\,] - \Pr_{g \stackrel{R}{\leftarrow} G}[\, A^g = 1\,]\,.$$

The probability is over the indicated random choice of g and the coin tosses of A. (This definition reflects the following intuition [GGM]. Consider the experiment in which A is provided as oracle a function g chosen at random from either F or from G, the choice being made at random according to a bit b. A is trying to predict b. The advantage is $2(\Pr[A^g = b] - 1/2)$, the amount that the probability that A is correct is bounded away from the guessing probability 1/2, scaled up to be between 0 and 1.)

Let F be a family with input length l and output length L, and R the family of random functions with the same parameters. We say that A $[t, q, \epsilon]$-breaks F if A runs in at most t steps, makes at most q oracle queries, and achieves $\mathsf{Adv}_A(F, R) \geq \epsilon$. The running time here is measured in a standard RAM model of computation.

Let family F have input length l and output length L, and let R be the family of random functions with the same parameters. To discuss security quantitatively, we say that statistical test A $[t, q, \epsilon]$-breaks F if A runs in at most t steps, makes at most q oracle queries, and achieves $\mathsf{Adv}_A(F, R) \geq \epsilon$. (The running time here is measured in a standard RAM model of computation.) In informal discussion, the finite function family F is said to be $[t, q, \epsilon]$-pseudorandom if there is no statistical test that $[t, q, \epsilon]$-breaks F. (To be fully formal one ought to consider also other parameters such as the "code size".) In other words, in time t and given q examples one cannot distinguish a random member of F from a random function with advantage more than ϵ.

Notice that the key size of the finite PRF family F does not need to be explicitly specified in the definition of security: its influence is captured in that it influences the values of t, q, ϵ for which the F is $[t, q, \epsilon]$-pseudorandom.

Note there is no security parameter. While, traditionally, all parameters mentioned would be considered functions of a security parameter k, for us they are numbers. Since we will be evaluating security exactly, the security parameter becomes unnecessary. It is still true that any scheme we specify is actually a uniform collection of schemes, but this is clear anyway and it is not worth introducing a parameter just to say this.

3.2 Message authentication

We provide formal definitions of schemes and their security in the exact security setting. We begin with stateless schemes, in which no counters or other state information need be maintained. Then we briefly indicate how the definitions should be updated to take account of state.

STATELESS SCHEMES. A *(stateless) message authentication scheme* consists of a *signing* algorithm Sig and a *verifying* algorithm Vf. The signing algorithm may be probabilistic; the verifying one typically is not. Associated to the scheme are parameters κ and L_{sig} describing the key length and MAC length, respectively. On input a κ-bit key a and a message M, algorithm Sig outputs an L_{sig}-bit string μ called the *signature*, or MAC, of M. On input a κ-bit key a, a message M and an L_{sig}-bit string μ, algorithm Vf outputs a bit, with 1 standing for accept and 0 for reject. We ask for a basic validity condition, namely that authentic signatures are accepted with probability one. That is, for any key a, message M, and signature μ which is output with positive probability by $\mathsf{Sig}(a, M)$, it must be the case that $\mathsf{Vf}(a, M, \mu) = 1$.

SECURITY. An adversary for a message authentication scheme is a probabilistic algorithm E which is given oracle access to the signer and verifier—more precisely, to $\mathsf{Sig}(a, \cdot)$ and $\mathsf{Vf}(a, \cdot, \cdot)$ for a random but hidden choice of a. E can request a signature of a message of her choice; to do this, she writes M on a special query tape. She can also ask the verifier to verify that μ is a valid signature for M; to do this she writes (M, μ) on a special query tape. Formally, *E's attack* on the scheme is described by the following experiment:

(1) A random string a of length κ is selected as the shared secret. A random string r_E is selected as the coin tosses of E. E now starts computing.

(2) Suppose E makes a signing query M. Then the oracle computes a signature $\mu \stackrel{R}{\leftarrow} \mathsf{Sig}(a, M)$ and returns it to E. (Since Sig may be probabilistic, this step requires making the necessary underlying choice of a random string for Sig, anew for each signing query.)

(3) Suppose E makes a verify query (M, μ). The oracle computes the decision $d = \mathsf{Vf}(a, M, \mu)$ and returns it to E.

The adversary is allowed an adaptive chosen message attack, as in the notion of [GMR], but we also allow verify queries because, unlike the setting in digital signatures, E cannot compute the verify predicate on her own (since the verify algorithm is not public). Note that E does not see a nor the coin tosses of Sig.

We say that E's attack on $\mathcal{M}$ is a (q_s, q_v)-attack if during the course of the attack she makes no more than q_s signing queries and no more than q_v verify queries. A (q_s, q_v)-attack is a (t, q_s, q_v)-attack if, in addition, E runs for no more than t steps, in the RAM model of computation we fixed above. It is useful to say that a verify query (M, μ) is *known-authentic* if a signing query M was made prior to this verify query and the signature returned was μ. Note validity implies that known-authentic verify queries are accepted. We thus assume of any adversary E that she never makes any known-authentic queries.

The outcome of running the protocol in the presence of an adversary is used to define security. We say that E is *successful* if she makes a verify query (M, μ) which is accepted but which is not known-authentic.[1] (The verify query (M, μ) in question is called a *forgery*, and the definition reflects the notion of existential forgery [GMR].) We say that E $[q_s, q_v, \epsilon]$-breaks the scheme if her attack is a (q_s, q_v, ϵ)-attack and her probability of success is at least ϵ. We say she $[t, q_s, q_v, \epsilon]$-breaks the scheme if her attack is a (t, q_s, q_v, ϵ)-attack and her probability of success is at least ϵ. In informal discussion we'll say the scheme is $[t, q_s, q_v, \epsilon]$-unforgeable if there is no adversary who can $[t, q_s, q_v, \epsilon]$-break it. (To be fully formal we would have to consider also other parameters like the "code size.")

STATEFUL SCHEMES. In a stateful message authentication scheme the signer maintains state across consecutive signing requests. (For example, in our counter-based scheme the signer maintains a message counter.) In such a case the signing algorithm can be thought of as taking an additional input —the "current" state C_s of the signer— and returning an additional output —the signer's next state. We must modify the experiment describing E's attack: in Step (1) we also have that C_s is initialized to a value specified by the scheme; and in Step (2), we compute $(\mu, C_s') \stackrel{R}{\leftarrow} \mathsf{Sig}(a, M, C_s)$, then return μ to the adversary and replace C_s by C_s'. Note the adversary doesn't see the revised state (though in the stateful scheme of this paper this wouldn't matter). Also note that, for simplicity, we allow the signer a state but not the verifier.

4 The Randomized XOR Scheme

We first present the general scheme, of which that in Section 2 is a special case, and then proceed to the security analysis.

4.1 Specification of the scheme

Let F be a family of functions with key length κ, input length l, and output length L. We fix in addition a parameter $b \leq l - 1$ which will be the block size. We will assume that any message M

[1] This is slightly stronger than the more standard definition in which one would only ask that the message M was not a previous signing query. We make this stronger requirement because we achieve it and because it is useful in contexts like entity authentication.

to be authenticated has length at most $|M| \leq b2^{l-b-1}$. By standard padding arguments we may assume wlog that the message length is a multiple of b. We then regard M as a sequence of b-bit blocks. The number of blocks is denoted $\|M\|_b$, and with b understood the i-th block is denoted $M[i]$, for $i = 1, \ldots, \|M\|_b$. Let $r \in \{0,1\}^{l-1}$, and let $a \in \{0,1\}^\kappa$ be the shared key. We define

$$\mathsf{tag}_{F,b}(a, M, r) \;=\; F_a(\,0\,.\,r\,) \;\oplus F_a(\,1\,.\,\langle 1\rangle_{l-b-1}\,.\,M[1]\,) \oplus \cdots \oplus F_a(\,1\,.\,\langle n\rangle_{l-b-1}\,.\,M[n]\,)\;. \qquad (1)$$

We'll use this function in both the randomized and the counter-based schemes. We'll call r the *seed.* The (stateless) message authentication scheme

function $\mathsf{SigR}_{F,b}(a, M)$	**function** $\mathsf{VfR}_{F,b}(a, M', (r', z'))$
$r \xleftarrow{R} \{0,1\}^{l-1}$; $z \leftarrow \mathsf{tag}_{F,b}(a, M, r)$	$z \leftarrow \mathsf{tag}_{F,b}(a, M', r')$
return (r, z)	**if** $z = z'$ **then return** 1 **else return** 0

We call $\mathsf{XMACR}_{F,b}$ the randomized XOR scheme based on function family F and using block size b. The validity condition is easy to verify. Note that the XMACR scheme of Section 2 is, in the current terminology, $\mathsf{XMACR}_{F,32}$ with F being the family specified by $F_a(\cdot)$ = first 48 bits of the output of $\mathrm{DES}_a(\cdot)$.

Trading efficiency for message length. Note that choosing different values of b will tradeoff the number of F_a computations with the allowable length of messages that can be signed. Namely, the scheme calls for $1 + \|M\|_b = 1 + (|M|/b)$ evaluations of F_a and allows $|M|$ to be $b2^{l-b-1}$ so that increasing b reduces the number of F_a evaluations at the cost of restricting the scheme to shorter messages. For example, the XMACR scheme of Section 2, with $b = 48$, currently has 1.33 times the number of DES operations of the CBC MAC, and allows $|M|$ up to $48 * 2^{15} = 3 * 2^{19}$. The latter is quite large. So we could further increase b, reducing the number of DES computations at the expense of decreasing the allowed length of M.

4.2 Security of the randomized XOR scheme

Information theoretic case. Begin by thinking of F as ideal (i.e., truly random). Namely, we consider $\mathsf{XMACR}_{R,b}$, which we call the information theoretic case. The following theorem provides an absolute bound on the success of the adversary in terms of the number of sign and verify queries she makes.

Theorem 4.1 Let R be the family of random functions with input length l and output length L, let b be at most $l-1$, and let E be any adversary making a (q_s, q_v)-attack on $\mathsf{XMACR}_{R,b}$. Then the probability that E is successful is at most $\delta_{\mathrm{R}} \stackrel{\mathrm{def}}{=} 2q_s^2 \cdot 2^{-l} + q_v \cdot 2^{-L}$.

Note the bound is independent of b: the latter figures only in our assumption that any query M made by E above satisfies $\|M\|_b \leq 2^{l-b-1}$.

The proof of Theorem 4.1 is given in Section A.1. It has two parts: first we relate the security of the scheme to the rank of an appropriate matrix; then we bound the rank of the matrix.

Computational case. We now assume we are given a family F which is not truly random, but $[t', q', \epsilon']$-pseudorandom. In that case, how secure is $\mathsf{XMACR}_{F,b}$? This is what the following tells us. It is the result of more direct interest in practice (although Theorem 4.1 is in some ways more basic). We show how to take an adversary E for $\mathsf{XMACR}_{F,b}$ and build from it an an adversary U_E which distinguishes F from a truly random functions. The construction is "uniform" in the sense that U_E is given by a fixed machine U with oracle access to E. The model "charges" U for its oracle queries whatever is their actual running time on E. The constant c below depends only on details of the computational model.

Theorem 4.2 There is an oracle machine U and a constant c such that the following is true. Let F be a family of functions with input length l and output length L and let b be at most $l-1$. Let E be an adversary who $[t, q_s, q_v, \epsilon]$-breaks $\mathsf{XMACR}_{F,b}$ and suppose any message M in a query of E has a number $\|M\|_b$ of blocks which is bounded by n. Let $\delta_{\mathrm{R}} = 2q_s^2 \cdot 2^{-l} + q_v \cdot 2^{-L}$. Then U^E $[t', q', \epsilon']$-breaks F, where

$$t' = t + c(l+L)q' \quad ; \quad q' = (q_s + q_v) \cdot (n+1) \quad ; \quad \epsilon' = \epsilon - \delta_{\mathrm{R}} .$$

In other words if F is $[t', q', \epsilon']$-pseudorandom (the values t', q', ϵ' depending on the key size and cryptanalytic strength of the finite PRF F) then $\mathsf{XMACR}_{F,b}$ is $[t, q_s, q_v, \epsilon]$-unforgeable, where $t = t' - c(l+L)q'$, $q_s + q_v = q'/(n+1)$ and $\epsilon = \epsilon' + \delta_{\mathrm{R}}$. Thus a success probability of δ_{R} for the adversary is unavoidable, even if the PRF is "ideal;" beyond that, the success of the adversary is bounded in terms of the parameters of the block cipher.

The proof of Theorem 4.2 is given in Section A.2.

4.3 Attacks (lower bounds)

Here we provide an attack on the randomized XOR scheme to show that the above analysis is essentially tight. Since we think of F as pseudorandom, we will do the attack assuming it is in fact random; that is, we look at $\mathsf{XMACR}_{R,b}$ where R is the family of random functions with input length l and output length L. Given q_s, q_v we specify a particular adversary E who makes q_s sign queries and q_v verify queries, and then outputs a forgery with probability $\epsilon \geq \Omega(\delta_{\mathrm{R}})$, where $\delta_{\mathrm{R}} = 2q_s^2 \cdot 2^{-l} + q_v \cdot 2^{-L}$. The proof of the following is in Appendix A.3.

Proposition 4.3 Let R be the family of random functions with input length l and output length L, and let b be at most $l-1$. Then there is a constant $c > 0$ such that for any q_s, q_v satisfying $q_s^2 \leq 2^l$ and $q_v \leq 2^L$, there is an adversary E who $[t, q_s, q_v, \epsilon]$-breaks $\mathsf{XMACR}_{R,b}$, where

$$t = c(l+L)(q_s + q_v) \quad ; \quad \epsilon = \max\left\{ \left(1 - \frac{1}{e}\right) \cdot \frac{q_s^2 - 3q_s}{2 \cdot 2^l} \, , \, \frac{q_v}{2^L} \right\} .$$

We remark that the proof actually shows more—that the adversary forges the signature of essentially any message of her choice. This makes the attack all the more relevant.

We also remark that being in the information theoretic setting we need not, strictly speaking, have provided the time of the adversary since she is in principle allowed to run for as long as she wants. We provide it to show that in fact the attack is practical; we aren't taking advantage of the leeway in the model.

5 The Counter-Based XOR Scheme

Here we present another scheme which enhances security by allowing the signer to maintain state in the form of a counter. The gain is greater security: the success probability of the adversary in the analogue of Theorem 4.1 does not depend on the number q_s of signing queries at all (as long as the latter is bounded by a certain exponential function of l)!

5.1 Specification of the counter-based scheme

Let F be a family of functions with key length κ, input length l, and output length L, and let the parameter b be as before. Let $a \in \{0,1\}^\kappa$ be the shared key. The idea is to use the the same tagging function as above, but use the current counter value as the seed. Formally the scheme $\mathsf{XMACC}_{F,b}$ is specified by functions $\mathsf{SigC}_{F,b}, \mathsf{VfC}_{F,b}$. The signing function depends on a counter C maintained by the signer; it is initially 0 and then incremented by the signing function itself. (In Section 2 we loosely identified C with its 63 bit representation. Now we are more precise, viewing it as an integer and writing $\langle C \rangle_{l-1}$ for the corresponding string.) The verifying function has no state. Below $\mathsf{tag}_{F,b}$ is the function specified in Equation 1 of Section 4.

function $\mathsf{SigC}_{F,b}(a, M, C)$	**function** $\mathsf{VfC}_{F,b}(a, M', (C', z'))$
$C \leftarrow C+1$; $z \leftarrow \mathsf{tag}_{F,b}(a, M, \langle C \rangle_{l-1})$	$z \leftarrow \mathsf{tag}_{F,b}(a, M', \langle C' \rangle_{l-1})$
return $((C, z),\ C)$	**if** $z = z'$ **then return** 1 **else return** 0

We call $\mathsf{XMACC}_{F,b}$ the counter-based XOR scheme based on function family F and using block size b. The validity of the counter-based XOR scheme is easy to verify. Note that the XMACC scheme of Section 2 is, in the current terminology, $\mathsf{XMACC}_{F,32}$ with F being the family specified by $F_a(\cdot)$ = first 48 bits of the output of $\mathrm{DES}_a(\cdot)$.

As before the length of any message whose signature the adversary requests is assumed bounded by $b2^{l-b-1}$. But also we will now assume that the total number of signing requests is bounded by 2^{l-1}. That is, we require that C not exceed 2^{l-1}. (Typically, this is not a significant restriction.) These assumptions are made in the theorems that follow.

5.2 Security of the counter-based scheme

In addition to the assumption that the length of any message whose signature the adversary requests is bounded by $b2^{l-b-1}$, we will now also assume that the total number of signing requests is bounded by 2^{l-1}. These assumptions are made in the theorems that follow.

Information theoretic case. The counter-based scheme dramatically increases the security as indicated below. The success probability of the adversary depends only on the number q_v of its verify queries, rather than this plus $2q_s^2 \cdot 2^{-l}$.

Theorem 5.1 Let R be the family of random functions with input length l and output length L, let b be at most $l-1$, and let E be any adversary making a (q_s, q_v)-attack on $\mathsf{XMACC}_{R,b}$, where $q_s < 2^{l-1}$. Then the probability that E is successful is at most $\delta_\mathrm{C} \stackrel{\mathrm{def}}{=} q_v \cdot 2^{-L}$.

To see concretely what this improvement means, think of F_a = first 48 bits of DES_a, where we have $l = 64$ and $L = 48$. If $q_s = 2^{20}$ and $q_v = 1$, then the success probability is a marginal 2^{-23} in the randomized scheme, but it is 2^{-48} in the counter-based one.

Computational case. We get a corresponding improvement:

Theorem 5.2 There is an oracle machine U and a constant c such that the following is true. Let F be a family of functions with input length l and output length L and let b be at most $l-1$. For $q_s < 2^{l-1}$ let E be an adversary who $[t, q_s, q_v, \epsilon]$-breaks $\mathsf{XMACC}_{F,b}$, and suppose any message M in a query of E has a number $\|M\|_b$ of blocks which is bounded by n. Let $\delta_\mathrm{C} = q_v \cdot 2^{-L}$. Then U^E $[t', q', \epsilon']$-breaks F, where

$$t' = t + c(l+L)q' \quad ; \quad q' = (q_s + q_v) \cdot (n+1) \quad ; \quad \epsilon' = \epsilon - \delta_\mathrm{C} \ .$$

Again, what this means is that if F is $[t, q', \epsilon']$-pseudorandom then $\mathsf{XMACC}_{F,b}$ is $[t, q_s, q_v, \epsilon]$-unforgeable, where $t = t' - c(l+L)q'$, $q_s + q_v = q'/(n+1)$, and, most importantly, $\epsilon = \epsilon' + \delta_{\mathrm{C}}$.

See Appendix A.4 for the proofs of the above theorems.

5.3 Attacks (lower bounds)

The best attack is just to guess signatures! Furthermore one does not expect better than the following (short of breaking the PRF) by virtue of the above theorems. The proof of the following is trivial, but provided in Appendix A.5 for the sake of completeness.

Proposition 5.3 Let R be the family of random functions with input length l and output length L, and let b be at most $l-1$. There is a constant $c > 0$ such that for any $q_v \leq 2^L$, there is an adversary E who $[t, 0, q_v, \epsilon]$-breaks $\mathsf{XMACC}_{R,b}$, where $t = c(l+L)q_v$ and $\epsilon = q_v \cdot 2^{-L}$.

Again, the adversary will in fact be forging the signature of a message of her choice.

6 A General Framework

The schemes we have presented above are instances of a very general framework which results in a class of block cipher based schemes whose security is reducible to a question on the rank of an associated matrix random variable. Here we describe this framework. We let F be a family of functions of input length l and output length L (to be thought of as pseudorandom). We let a denote the key shared between signer and verifier. We call our schemes *XOR schemes*. Below we consider only stateless schemes; stateful ones can be treated similarly.

XOR SCHEMES. A (stateless) XOR scheme is specified by a pair of functions $\mathcal{R}$ and $\mathcal{E}$ called the *randomizing* and *encoding* functions, respectively. The first function $\mathcal{R}$ is probabilistic, and when applied to a message M produces a string r.[2] The second function $\mathcal{E}$ is deterministic, and when applied to M, r produces a set D of l-bit strings. The signing and verifying functions are:

function $\mathsf{SigG}_{F,\mathcal{R},\mathcal{E}}(a, M)$	**function** $\mathsf{VfG}_{F,\mathcal{R},\mathcal{E}}(a, M', (r', z'))$
$\quad r \xleftarrow{R} \mathcal{R}(M)$; $D \leftarrow \mathcal{E}(M, r)$	$\quad D \leftarrow \mathcal{E}(M', r')$
$\quad z \leftarrow \bigoplus_{x \in D} F_a(x)$	$\quad z \leftarrow \bigoplus_{x \in D} F_a(x)$
$\quad$ **return** (r, z)	$\quad$ **if** $z = z'$ **then return** 1 **else return** 0

We denote by $\mathsf{XMACR}_{F,\mathcal{R},\mathcal{E}}$ the message authentication scheme consisting of the signing and verifying functions above.

RECOVERING PREVIOUS SCHEMES. This does indeed generalize our previous schemes; for example, $\mathsf{XMACR}_{F,b}$ is $\mathsf{XMACR}_{F,\mathcal{R},\mathcal{E}}$ for $\mathcal{R}$ being the function which given M outputs a random $l-1$ bit string r, and $\mathcal{E}$ being the function which given M and r outputs the set $D = \{0 \,.\, r\} \cup \{1 \,.\, \langle i \rangle_{l-b-1} \,.\, M[i] : i = 1, \ldots, \|M\|_b\}$.

SECURITY. We now discuss the security of a general XOR scheme $\mathsf{XMACR}_{F,\mathcal{R},\mathcal{E}}$. The main issue is the information theoretic case. For this let R be the family of random functions with input length l and output length L, and let E be an optimal (wlog deterministic) adversary. Let $\mathsf{M}_1, \ldots, \mathsf{M}_{q_s}$ be the random variables which are the signing queries made by E, and let $\mathsf{R}_i = \mathcal{R}(\mathsf{M}_i)$ for $i = 1, \ldots, q_s$. Let

[2] In the simplest case $\mathcal{R}$ depends only on M, but we can allow it to depend on previous messages, their signatures, and even previous coin tosses of the sender. However we don't allow it to depend on the key a.

$\mathsf{D}_i = \mathcal{E}(\mathsf{M}_i, \mathsf{R}_i) \subseteq \{0,1\}^l$ and let A_i be the 2^l-bit characteristic vector of the set D_i, for $i = 1, \ldots, q_s$. Also let M be a message and r a string, and let A_{q_s+1} be the characteristic vector of $\mathcal{E}(M, r)$. We let $\textsc{Matrix}_{q_s}(M, r)$ denote the random variable which is the $q_s + 1$ rows and 2^l columns matrix whose i-th row is A_i, for $i = 1, \ldots, q_s + 1$. Now let

$$\mathsf{NFRank}_{q_s}(M, r) \;=\; \Pr\left[\, \textsc{Matrix}_{q_s}(M, r) \text{ doesn't have full rank} \mid M \notin \{\mathsf{M}_1, \ldots, \mathsf{M}_{q_s}\} \,\right]$$

denote the probability that the matrix is not of full rank given that M was not a signing query. The probability is over the coin tosses of the signer (namely the coin tosses of $\mathcal{R}$) and initial choice of a determining the function R_a used by the signer.

Theorem 6.1 Let R be the family of random functions with input length l and output length L. Then the probability that E is successful in a (q_s, q_v)-attack on $\mathsf{XMACR}_{R,\mathcal{R},\mathcal{E}}$ is at most

$$q_v \cdot 2^{-L} + \max_{M,r} \{\mathsf{NFRank}_{q_s}(M, r)\} \,.$$

The proof uses ideas from the proof of Theorem 4.1. The computational analogue can be similarly derived.

7 Comparison with the CBC MAC

We compare the security of our schemes to that of the CBC MAC. First, let us recall that scheme. Let F be a family of functions with input and output length l. A message $M = M[1] \ldots M[n]$ is viewed as a sequence of l-bit blocks. The (full) CBC scheme is specified by the following:

function $\mathsf{SigCBC}_{F,n}(a, M[1] \ldots M[n])$	**function** $\mathsf{VfCBC}_{F,n}(a, M', \mu')$
$\quad y_0 \leftarrow 0^l$	$\quad \mu \leftarrow \mathsf{SigCBC}_{F,n}(a, M')$
$\quad$ **for** $i \leftarrow 1$ **to** n **do** $y_i \leftarrow F_a(\, y_{i-1} \oplus M[i] \,)$	$\quad$ **if** $\mu = \mu'$ **then return** 1
$\quad$ **return** y_n	$\qquad\qquad$ **else return** 0

The scheme is denoted $\mathsf{CBC\text{-}MAC}_{F,n}$. We will consider the information theoretic case. The following was proved in [BKR]. Let R be the family of random functions of input and output length l, and let E be any adversary. Then the probability that E $[q_s, q_v, \epsilon]$-breaks $\mathsf{CBC\text{-}MAC}_{R,n}$ is at most $\delta_{\mathrm{CBC}} \stackrel{\mathrm{def}}{=} 3(n^2+1) \cdot (q_s + q_v)^2 \cdot 2^{-l}$. To compare this to our schemes set $L = l$ in Theorems 4.1 and 5.1. Clearly, δ_{R} is smaller than δ_{CBC}, and δ_{C} is considerably smaller than δ_{CBC}; in particular, δ_{R} and δ_{C} don't depend on n while δ_{CBC} does, a significant difference.

Yet this by itself is not proof that our schemes are more secure, because it may by that the analysis of [BKR] is not tight. In fact, however, there are attacks (lower bounds) which indicate that the best improvement one could hope for in their analysis would be that $\delta_{\mathrm{CBC}} = \Omega(nq_s^2 + q_v)2^{-l}$. This result is due independently to Krawczyk [Kr] and Preneel and Van Oorschot [PV]— what they show is an attack on the CBC MAC which succeeds in forging the signature of a new message with probability $\Omega(nq_s^2) \cdot 2^{-l}$, after having made q_s signing queries on n-block messages. Thus the dependence on n in δ_{CBC} is unavoidable.

We comment that the $\mathsf{CBC\text{-}MAC}_{F,n}$ is only secure for fixed n; the scheme must be modified to accommodate n's of varying length. In contrast, both $\mathsf{XMACR}_{F,b}$ and $\mathsf{XMACC}_{F,b}$ operate on inputs of varying lengths (with the security bounds given by our theorems).

Acknowledgments

We thank Mike Luby for pointing out an oversight in a previous version of the proof of Theorem 4.1. We thank Hugo Krawczyk for a great deal of useful advice and information. We thank Marc Fischlin for his careful reading and many useful comments and corrections.

References

[ABI] N. Alon, L. Babai and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *J. of Algorithms*, Vol.7, 567-583, 1986.

[BKR] M. Bellare, J. Kilian and P. Rogaway. On the security of cipher block chaining. *Advances in Cryptology – Crypto 94 Proceedings*, Lecture Notes in Computer Science Vol. 839, Y. Desmedt ed., Springer-Verlag, 1994.

[BGG1] M. Bellare, O. Goldreich and S. Goldwasser. Incremental cryptography: The case of hashing and signing. *Advances in Cryptology – Crypto 94 Proceedings*, Lecture Notes in Computer Science Vol. 839, Y. Desmedt ed., Springer-Verlag, 1994.

[BGG2] M. Bellare, O. Goldreich and S. Goldwasser. Incremental cryptography and application to virus protection. Proceedings *of the* 27th *Annual Symposium on Theory of Computing*, ACM, 1995.

[BeRo] B. Berger and J. Rompel, "Simulating ($\log^c n$)-wise independence in NC," Proceedings *of the* 30th *Symposium on Foundations of Computer Science*, IEEE, 1989.

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

[GMR] 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.

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

[Kr] H. Krawczyk. Personal communication, September 1994.

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

[PV] B. Preneel and P. van Oorschot. A new generic attack on message authentication codes. *Advances in Cryptology – Crypto 95 Proceedings*, Lecture Notes in Computer Science Vol. 963, D. Coppersmith ed., Springer-Verlag, 1995.

[Ts] G. Tsudik, "Message authentication with one-way hash functions." *Proceedings of Infocom 92,* IEEE Press, 1992.

[Ri] R. Rivest, "The MD5 message digest algorithm." IETF RFC-1321, 1992.

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

A Proofs of Theorems and Propositions

We provide here the proofs for the theorems in Section 4.2 and Section 5.2 as well as the Propositions in Section 4.3 and Section 5.3.

A.1 Proof of Theorem 4.1

Let R be the family of random functions with input length l and output length L. Since E is computationally unbounded we may assume wlog that it is deterministic. The probabilistic choices in E's attack on the scheme are thus the initial choice a of key (naming a random function $R_a \in R$), and the choices of seeds made by the signer in the course of signing. We may assume $q_s < 2^{l-1}$ (wlog because otherwise there is nothing to prove) and that E makes exactly q_s signing queries and exactly q_v verify queries.

Let M_i denote the random variable whose value is the i-th message whose signature E requests. Let R_i denote the random variable whose value is the random seed chosen by the signer to sign M_i and let $\mathsf{Z}_i = \mathsf{tag}_{R,b}(a, \mathsf{M}_i, \mathsf{R}_i)$ denote its tag, $i = 1, \ldots, q_s$. Let $\mathsf{Distinct}$ be the event that $\mathsf{R}_1, \ldots, \mathsf{R}_{q_s}$ are all distinct and Succ the event that E is successful.

Fact A.1 Let $P(m,t)$ denote the probability of at least one collision in the experiment of throwing t balls, independently at random, into m buckets. Then $P(m,t) \le t^2/m$.

Remark A.2 If $t \le m/2$ we have the slightly better bound $P(m,t) \le 1 - e^{-(t^2-t)/m}$. (It can be derived from the standard birthday calculation by using the fact that $1-x > e^{-2x}$ for $0 < x \le 1/2$, which in turn can be derived from Fact A.5 applied to $2x$.) But we won't use this.

Continuing the proof, we have

$$\begin{aligned} \Pr[\,\mathsf{Succ}\,] &\le \Pr[\,\mathsf{Succ} \mid \mathsf{Distinct}\,] + \Pr[\,\neg\mathsf{Distinct}\,] \\ &= \Pr[\,\mathsf{Succ} \mid \mathsf{Distinct}\,] + P(2^{l-1}, q_s)\ . \end{aligned}$$

Using the above fact, the second term above is at most $q_s^2/2^{l-1} = 2q_s^2 \cdot 2^{-l}$. Below we will show that

$$\Pr[\,\mathsf{Succ} \mid \mathsf{Distinct}\,] \le q_v \cdot 2^{-L}\ , \tag{2}$$

whence the theorem follows.

Now fix a particular sequence of messages $M_1, \ldots, M_{q_s}$, a particular choice $r_1, \ldots, r_{q_s} \in \{0,1\}^{l-1}$ of *distinct* seeds and a particular choice $z_1, \ldots, z_{q_s}$ of L-bit strings, for which

$$\Pr[\,\mathsf{M}_i = M_i \text{ and } \mathsf{R}_i = r_i \text{ and } \mathsf{Z}_i = z_i \text{ for } i = 1, \ldots, q_s\,] > 0\ . \tag{3}$$

We let

$$\Pr_1[\,\cdot\,] = \Pr[\,\cdot \mid \mathsf{M}_i = M_i \text{ and } \mathsf{R}_i = r_i \text{ and } \mathsf{Z}_i = z_i \text{ for } i = 1, \ldots, q_s\,]$$

denote the probability conditioned upon E's having requested the specified messages, the signer having chosen the specified random strings in the signing process, and the tags returned being the specified strings. (The probability is effectively over only the random choice of the shared key a, since everything else is fixed.) Below we will show that

$$\Pr_1[\,\mathsf{Succ}\,] \le q_v \cdot 2^{-L}\ . \tag{4}$$

Since $M_1, \ldots, M_{q_s}$, $r_1, \ldots, r_{q_s}$ and $z_1, \ldots, z_{q_s}$ were arbitrary, standard conditioning arguments can be used to obtain Equation 2. (Note that E's queries are adaptive, so that M_i depends on $\mathsf{Z}_1, \ldots, \mathsf{Z}_{i-1}$. This was the reason to condition also on values of $\mathsf{Z}_1, \ldots, \mathsf{Z}_{q_s}$.)

$$\begin{array}{rc}
 & \begin{array}{ccccccccccccc} & & & \alpha & & & & & & & \beta & & \\ & & & \downarrow & & & & & & & \downarrow & & \end{array} \\
\begin{array}{r} \\ \\ \\ \alpha \ \rightarrow \\ \\ q_s+1 \ \rightarrow \end{array} &
\left[\begin{array}{cccccccc|ccccc}
1 & & & & & 0 & \cdots & 0 & * & \cdots & * & \cdots & * \\
 & 1 & & & & \vdots & & \vdots & * & \cdots & * & \cdots & * \\
 & & 1 & & & \vdots & & \vdots & * & \cdots & * & \cdots & * \\
 & & & 1 & & \vdots & & \vdots & * & \cdots & * & \cdots & * \\
 & & & & 1 & 0 & \cdots & 0 & * & \cdots & * & \cdots & * \\
\hline
0 & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 & * & \cdots & * & \cdots & *
\end{array}\right]
\end{array}$$

Figure 1: *The matrix B for Case 2 of the proof of Lemma A.3. This example has $q_s = 5$ and $\alpha = 4$.*

In what follows, we make the simplifying assumption that E first makes its q_s signing queries, and then makes its q_v verify queries. Later we will see how to cover the case that verify queries are interspersed with signing queries.

Fix a message M_{q_s+1} distinct from $M_1, \ldots, M_{q_s}$, a seed $r_{q_s+1} \in \{0,1\}^{l-1}$ and an L-bit string z_{q_s+1}. These are intended to stand for a possible forgery $(M_{q_s+1}, (r_{q_s+1}, z_{q_s+1}))$. Notice that although M_{q_s+1} is distinct from previous messages, r_{q_s+1} is not assumed distinct from previous seeds– indeed, since the adversary may choose it, we cannot make such an assumption. Below we will show that

$$\Pr_1\left[\,\mathsf{tag}_{R,b}(a, M_{q_s+1}, r_{q_s+1}) = z_{q_s+1}\,\right] \;\leq\; 2^{-L}\ . \tag{5}$$

This means that were E to make the verify query $(M_{q_s+1}, (r_{q_s+1}, z_{q_s+1}))$, having previously queried $M_i, \ldots, M_{q_s}$ and got back $(r_1, z_1), \ldots, (r_{q_s}, z_{q_s})$ in reply, her success probability would be at most 2^{-L}. But our choice of $M_{q_s+1}, r_{q_s+1}, z_{q_s+1}$ was arbitrary, and the number of verify queries that E can make is at most q_v. Using conditioning arguments one can obtain Equation 4. Thus the main claim is Equation 5 and what follows is devoted to its proof.

Recall $M_i[j] \in \{0,1\}^b$ denotes the j-th block of M_i. We define a q_s+1 by 2^l matrix B over GF(2). Its rows are indexed $1, \ldots, q_s+1$ and its columns are indexed by the l-bit strings in lexicographic order. The entry in row i, column x is denoted $B[i,x]$, and is defined as follows. First consider the case where the first bit of x is 0, so that $x = 0\,.\,y$. Then we set $B[i,x] = 1$ if $y = r_i$ and 0 otherwise. Now suppose the first bit of x is 1, and write it as $x = 1\,.\,\langle j\rangle_{l-b-1}\,.\,y$, where $|y| = b$. Then we set $B[i,x] = 1$ if $M_i[j] = y$ and 0 otherwise. (In particular, $B[i,x] = 0$ if $j > \|M_i\|_b$.) Note the matrix is not a random variable—it is fixed given that $M_1, \ldots, M_{q_s+1}$ and $r_1, \ldots, r_{q_s+1}$ are fixed.

Lemma A.3 The matrix B has full rank.

Proof: We will transform B by row and column operations until it has a q_s+1 by q_s+1 identity matrix in its upper left corner. At any time, the left half of B means the first 2^{l-1} columns and the right half means the rest.

In our initial matrix B, the left half consists of those columns whose index has first bit 0, and the right half consists of those columns whose index has first bit 1. Since $r_1, \ldots, r_{q_s}$ are distinct, each of rows $i = 1, \ldots, q_s$ has exactly one 1 in its left half. Thus we can permute columns until the first q_s rows of the left half of B consist of a q_s by q_s identity matrix followed by a q_s by $2^{l-1} - q_s$ matrix of zeroes. We now consider two cases.

Case 1. r_{q_s+1} is distinct from $r_1, \ldots, r_{q_s}$.

In this case, the last row of B has exactly one 1 in its left half, and this 1 is in a column otherwise zero. A single column swap extends our identity matrix by one, so that B is seen to have rank $q_s + 1$ as desired.

Case 2. $r_{q_s+1} = r_\alpha$ for some $\alpha \in \{1, \ldots, q_s\}$.

(An example corresponding to this case is in Figure 1.) Note that since $r_1, \ldots, r_{q_s}$ are distinct, α is unique. So the left half of row $q_s + 1$ consists of a 1 in position α and zeros elsewhere.

Add row α to row $q_s + 1$. This makes the left half of row $q_s + 1$ entirely 0. On the other hand, since M is by assumption different from M_α, the right halves of rows α and $q_s + 1$ are different; thus their sum has a 1 in some column $\beta \in \{2^{l-1} + 1, \ldots, 2^l\}$, so row $q_s + 1$ now has a 1 in column β. Any ones in rows $1, \ldots, q_s$ of column β can be zeroed out, specifically by adding column i to column β for any $i \in \{1, \ldots, q_s\}$ such that $B[i, \beta] = 1$. Finally, swap columns $q_s + 1$ and β. This results in a $q_s + 1$ by $q_s + 1$ identity matrix in the upper left corner of B as desired. ■

We now establish Equation 5. Similar relations of linear to probabilistic independence have been used in several places, for example [ABI, BeRo]. Let $W = \{0,1\}^{2^l}$ and regard elements of W as 2^l-bit vectors over GF(2). Identify a key a describing the function R_a with an L-tuple $(w_1, \ldots, w_L) \in W^L$. The value of the corresponding function at $x \in \{0,1\}^l$ is the L-bit string $w_1^{(x)} \ldots w_L^{(x)}$ formed by taking the x-th component of each vector. Identifying a with $(w_1, \ldots, w_L)$ in this way, notice that for each $j = 1, \ldots, L$ it is the case that $z_i^{(j)}$ is the dot product of the i-th row of B with the vector w_j, for $i = 1, \ldots, q_s$. Now let A be the matrix consisting of the first q_s rows of B. Also for $j = 1, \ldots, L$ let

$$u_j = \begin{bmatrix} z_1^{(j)} \\ \vdots \\ z_{q_s}^{(j)} \end{bmatrix} \quad \text{and} \quad v_j = \begin{bmatrix} z_1^{(j)} \\ \vdots \\ z_{q_s}^{(j)} \\ z_{q_s+1}^{(j)} \end{bmatrix}$$

Then observe that

$$\Pr_1\left[\,\mathsf{tag}_{R,b}(a, M_{q_s+1}, r_{q_s+1}) = z_{q_s+1}\,\right] = \frac{|B^*|}{|A^*|} \tag{6}$$

where

$$\begin{aligned} A^* &= \{\, (w_1, \ldots, w_L) \in W^L \;:\; Aw_j = u_j \text{ for } j = 1, \ldots, L \,\} \\ B^* &= \{\, (w_1, \ldots, w_L) \in W^L \;:\; Bw_j = v_j \text{ for } j = 1, \ldots, L \,\}\,. \end{aligned}$$

We fix the following notation: $A_* = \{0,1\}^{2^l - q_s}$, and $B_* = \{0,1\}^{2^l - q_s - 1}$. As before, regard elements of A_* as $(2^l - q_s)$-bit vectors over GF(2), and regard elements of B_* as $(2^l - q_s - 1)$-bit vectors over GF(2). Since B has full rank, it can be extended to a non-singular 2^l by 2^l matrix C. Let $C^{(i \ldots j)}$ denote the matrix consisting of rows i through j of C. The non-singularity of C implies that the map of B^* to B_*^L given by

$$(w_1, \ldots, w_L) \mapsto (\, C^{(q_s+2 \ldots 2^l)} w_1, \ldots, C^{(q_s+2 \ldots 2^l)} w_L \,)$$

is a bijection. Thus $|B^*| = |B_*^L| = 2^{L(2^l - q_s - 1)}$. Similarly $|A^*| = |A_*^L| = 2^{L(2^l - q_s)}$. So $|B^*|/|A^*| = 2^{-L}$. Now apply Equation 6 to obtain Equation 5.

Finally, we need to address the assumption made above that E first made all its q_s signing queries and then later made its q_v verify queries. We provide only a sketch of how to handle this.

Given an adversary E' who interspersed verify and sign queries, construct the following adversary E. E runs E', and whenever E' makes a verify query, it (1) records this query; (2) returns an answer of 0; (3) then continues to run E'. (Recall that we have assumed E makes no known-authentic oracle queries.) Finally when E' has completed her signing queries, E makes all the saved up verify queries. Now consider the first verify query of E'. If it was successful, so would be the first verify query of E. On the other hand if it failed, then E is correctly simulating E' in the sequel. Now apply the same argument to the second verify query, and so on. This means that the success probability of E' is bounded above by that of E, and we can apply the above to conclude.

A.2 Proof of Theorem 4.2

We may assume $\epsilon > \delta_{\mathrm{R}} = 2q_s^2 \cdot 2^{-l} + q_v \cdot 2^{-L}$ since otherwise there is nothing to prove. Let R be the family of random functions with input length l and output length L. Let E be an adversary. We now specify an algorithm A_E which has oracle access to a function g: $\{0,1\}^l \to \{0,1\}^L$, and is trying to decide whether this function is from F or from R.

For notational simplicity we'll assume that any message M whose signature E requests has $\|M\|_b = n$. It is convenient to let

$$\mathsf{tag}(M,r) \;=\; g(\,0\,.\,r\,) \oplus g(\,1\,.\,\langle 1\rangle_{l-b-1}\,.\,M[1]\,) \oplus \;\cdots\; \oplus g(\,1\,.\,\langle n\rangle_{l-b-1}\,.\,M[n]\,)\;.$$

Note that computing $\mathsf{tag}(M,r)$ requires A_E to make $n+1$ oracle queries. Algorithm A_E operates as follows.

- A_E selects a random string r_E as the coin tosses of E, and starts running E.
- Suppose E requests a signature of some message M. Then A_E picks $r \in \{0,1\}^{l-1}$ at random, computes $z = \mathsf{tag}(M,r)$, and returns (r,z) to E.
- Suppose E requests verification of some (M,μ) where $\mu = (r,z)$. Then A_E returns 1 if $z = \mathsf{tag}(M,r)$ and 0 otherwise.

If E is successful (which A_E can determine because A_E is answering all verify queries) then A_E outputs 1; else A_E outputs 0.

Clearly A_E makes at most $q' = (q_s + q_v)\cdot(n+1)$ oracle queries. It is easy to check that there is an oracle machine U and a constant c such that $U^E = A_E$ and the running time of A_E is bounded by $t + clq'$, for any E. Now let

$$\epsilon_1 \;=\; \Pr_{g \xleftarrow{R} F}[\,A_E^g = 1\,] \qquad \text{and} \qquad \epsilon_2 \;=\; \Pr_{g \xleftarrow{R} R}[\,A_E^g = 1\,]\;.$$

We leave to the reader to check that $\epsilon_1 = \epsilon$. Theorem 4.1 implies that $\epsilon_2 \le \delta_{\mathrm{R}}$. Thus

$$\mathsf{Adv}_{A_E}(F,R) \;=\; \epsilon_1 - \epsilon_2 \;\ge\; \epsilon - \delta_{\mathrm{R}}\;.$$

This completes the proof of Theorem 4.2.

A.3 Proof of Proposition 4.3

Actually we prove something a little stronger, namely that the adversary can forge the signature of almost any message of her choice. Specifically, suppose A', B' are distinct b-bit strings; we will show how to forge the signature of the message $M_4 = A'\,.\,B'$. (The extension to messages of more than two blocks is simple and is omitted.)

The adversary E chooses a b-bit string $A \notin \{A', B'\}$ and another b bit string $B \notin \{A', B', A\}$. She then sets $M_1 = A\,.\,B$, $M_2 = A'\,.\,B$ and $M_3 = A\,.\,B'$. She also sets $q = \lfloor (q_s - 1)/2 \rfloor$. Now she mounts the following attack–

(1) For $i = 1,2$, she makes the signing query M_i a total of q times. Let $(r_{i,j}, z_{i,j})$ denote the answers, $i = 1,2$ and $j = 1, \ldots, q$.

(2) She makes the signing query M_3. Let (r, z_3) denote the answer.

Notice that the total number of signing queries made is $2q + 1 \le q_s$. Now let Coll be the event that there exist j_1, j_2 such that $r_{1,j_1} = r_{2,j_2}$. Then:

(3) If Coll is true then E sets $\mu = (r, z)$ where $z = z_{1,j_1} \oplus z_{2,j_2} \oplus z_3$. She then makes the verify query (M_4, μ).

(4) Else, she picks a random $r' \in \{0,1\}^{l-1}$ and lets $z'_1, \ldots, z'_{q_v}$ be distinct L-bit strings, for example the q_v lexicographically least L-bit strings. She makes the q_v verify queries $(M_4, (r', z'_j))$ for $j = 1, \ldots, q_v$.

Note the number of verify queries made is at most q_v. For the analysis, first assume Coll is not true. Then E executes Step (4). But the tag $R_a(0 \,.\, r') \oplus R_a(1 \,.\, \langle 1 \rangle_{l-b-1} \,.\, A') \oplus R_a(1 \,.\, \langle 2 \rangle_{l-b-1} \,.\, B')$ is uniformly distributed, so clearly E is successful with probability $q_v \cdot 2^{-L}$. Now note that if Coll is true then

$$R_a(0 \,.\, r) \oplus R_a(1 \,.\, \langle 1 \rangle_{l-b-1} \,.\, A') \oplus R_a(1 \,.\, \langle 2 \rangle_{l-b-1} \,.\, B') \;=\; z \;.$$

Thus (r, z) is a valid MAC of M_4, meaning E is successful if Coll is true. We now want to lower bound the probability of Coll. We will need the following counterpart of Fact A.1 which provides a lower bound on the same quantity.

Fact A.4 Let $P(m,t)$ denote the probability of at least one collision in the experiment of throwing t balls, independently at random, into m buckets. Then $P(m,t) \ge 1 - e^{-(t^2-t)/(2m)}$.

To estimate the above we will use the following.

Fact A.5 If $0 \le x \le 1$ then $1 - e^{-x} \ge \left(1 - \frac{1}{e}\right) \cdot x$.

Proof: Consider the function $f_\alpha\colon [0,1] \to \mathsf{R}$ defined by $f_\alpha(x) = 1 - e^{-x} - xe^{-\alpha}$, where $\alpha \in [0,1]$ is a parameter which we want to choose so that $f_\alpha \ge 0$ on its entire domain $[0,1]$. First note that $f_\alpha(0) = 0$. Now, the derivative of f_α is $f'_\alpha(x) = e^{-x} - e^{-\alpha}$. Thus $f'_\alpha(x) \ge 0$ for $x \in [0, \alpha]$ and $f'_\alpha(x) \le 0$ for $x \in [\alpha, 1]$. This means f_α increases as x goes from 0 to α, then decreases. Thus, if we guarantee that $f_\alpha(1) \ge 0$ it follows that $f_\alpha \ge 0$ on $[0,1]$, on desired. But $f_\alpha(1) = 1 - (1/e) - e^{-\alpha}$. Thus setting $e^{-\alpha} = 1 - (1/e)$ does indeed guarantee $f_\alpha(1) \ge 0$, concluding the proof. ■

By symmetry $\Pr[\,\mathsf{Coll}\,] = (1/2) \cdot P(2^{l-1}, 2q)$. Let $x = [(2q)^2 - 2q] \cdot 2^{-l}$. Note $q \le (q_s/2) - (1/2)$ by choice of q, whence our assumption $q_s^2 \le 2^l$ implies $x \le 1$. Now use the above facts to see that $\Pr[\,\mathsf{Coll}\,]$ is at least

$$\left(1 - \frac{1}{e}\right) \cdot \frac{x}{2} \;\ge\; \left(1 - \frac{1}{e}\right) \cdot \frac{(2q)^2 - 2q}{2 \cdot 2^l} \;.$$

But $(q_s/2) - 1 \le q \le (q_s/2) - (1/2)$ so $(2q)^2 - 2q \ge (q_s - 2)^2 - (q_s - 1) \ge q_s^2 - 3q_s$. Thus

$$\Pr[\,\mathsf{Coll}\,] \;\ge\; \left(1 - \frac{1}{e}\right) \cdot \frac{q_s^2 - 3q_s}{2 \cdot 2^l} \;.$$

Thus the success probability ϵ of E is indeed at least the maximum of the above and $q_v \cdot 2^{-L}$.

A.4 Proofs of Theorems 5.1 and 5.2

It is easy to see what the counter buys us. In terms of the proof of Theorem 4.1 in Appendix A.1 above, we can think as though $\Pr[\,\mathsf{Distinct}\,] = 1$, because the counter values, now playing the

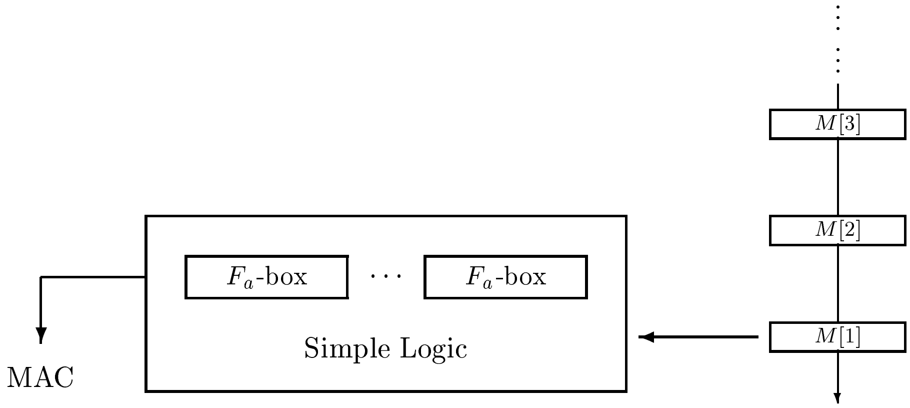

Figure 2: *The first message block is being acted on by the hardware. In another b/ν_N seconds, the second message block will be acted on. The signature is produced after all blocks are processed.*

roles of seeds, are distinct by definition, given that $q_s < 2^{l-1}$. Now Equation 2 can be argued as before, and $\Pr[\,\mathsf{Succ}\,]$ is bounded by just this. The proof of Theorem 5.2 is just like the proof of Theorem 4.2 in Appendix A.2 above.

A.5 Proof of Proposition 5.3

Let M be any message. We show how to forge its signature. The adversary sets $C = 1$. She then picks q_v distinct L-bit strings $z_1, \ldots, z_{q_v}$, for example the lexicographically least ones. She makes no sign queries. She just makes the q_v verify queries $(M_4, (C, z_j'))$ for $j = 1, \ldots, q_v$. Since $\mathsf{tag}_{R,b}(a, M, C)$ is uniformly distributed (a being the shared key used for the scheme) her success probability is clearly $q_v \cdot 2^{-L}$.

B The high-speed network setting

XOR schemes are particularly useful for message authentication over high speed networks. Here we describe the problem in this setting in more detail.

See Figure 2. The message M comes to the signer down a wire, at the rate of ν_N bits per second ("N" for "network"). We visualize the message as a sequence of b-bit blocks (buffering, for example, to create this illusion) so that we view ourselves as getting a sequence of b-bit blocks (at a rate of ν_N/b blocks per second). We assume the last block $M[n]$ is followed by some sort of marker (or is otherwise distinguished) so that the signer knows when the message is over.

The signer has available some hardware—simple logic, plus some reasonable number of chips to compute F_a ("F_a-boxes"), plus some fixed amount of memory. The amount of this hardware is fixed, independent of n. In particular, the message may be long and the signer does not have enough memory to store it. Nor can the signer see unrecorded bits once they've gone by. Rather, the MAC computation must be "on-line" in the following sense.

Let the memory have some initial content y_0 (this value may be computed by the hardware). Now, upon receiving $M[1]$ the signer computes some function G of y_0 and $M[1]$. (G is specified by the hardware, and the hardware can compute it in the time b/ν_N between block arrivals.) Call the result y_1. This value is written to memory, and replaces the value of y_0 that used to be there. Upon receiving $M[2]$, the same function G is applied to y_1 and $M[2]$ to compute a value y_2 which replaces y_1. And so on. After n steps the memory contains a value y_n. A little more processing is allowed (the number of steps is the latency λ of the scheme, and must be fixed and independent of n). Then the signer must output the MAC of $M = M[1] \cdots M[n]$.

Suppose an F_a-box computes at a rate of ν_F bits per second (ie. $F_a(x)$ is known $l\nu_F$ seconds after x is presented to the box). The constant ν_F is determined by the underlying chip technology. Let $\omega = \nu_N/\nu_F$; this is the factor by which the network is faster than the cryptographic hardware. If $\omega \leq 1$ (the F_a-boxes can "keep pace" with the network bandwidth) then $\mathsf{SigCBC}_{F,n}$ is a good solution to our problem: set $b = l = L$; set the initial content y_0 of the memory the memory to 0^l; the function G is $G(y,x) = F_a(y) \oplus x$; and $\mathrm{MAC} = y_n$ is the output.

On the other hand, suppose $\omega > 1$. Then we can't compute F_a in the interval of time between block arrivals. For example, say the ratio is two. In order to "keep up" with the network we could try to set two F_a-computations off and running in parallel. We have enough F_a-boxes, but *the scheme itself must admit parallelism* if the extra F_a boxes are going to help. For $\mathsf{SigCBC}_{F,n}$, the extra hardware is useless.

Technological evolution has made $\omega > 1$ on modern Gigabit-networks. Furthermore, this value continues to increase: advances in communications technology are outpacing speed increases of cryptographic hardware.

In our scheme, all the computations of F_a required to get z can be made in parallel, and the final results need only to be XORed to get z. This is what we mean by the scheme being "fully parallelizable." One can check that furthermore, the parallelizability is on-line: using $p = \omega l/b$ F_a-boxes we can compute $\mathsf{SigR}_{F,b}$ at a rate which enables us to keep up with the network. Thus, we can arbitrarily match network bandwidth by adding additional hardware, something which was impossible with $\mathsf{SigCBC}_{F,n}$.