Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC

November 3, 2003

Abstract

We describe highly efficient constructions, XE and XEX, that turn a blockcipher E: $\mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ into a tweakable blockcipher $\widetilde{E}$: $\mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ having tweak space $\mathcal{T} = \{0,1\}^n \times \mathbb{I}$ where $\mathbb{I}$ is a set of tuples of integers such as $\mathbb{I} = [1\,..\,2^{n/2}] \times [0\,..\,10]$. When tweak T is obtained from tweak S by incrementing one if its numerical components, the cost to compute $\widetilde{E}_K^T(M)$ having already computed some $\widetilde{E}_K^S(M')$ is one blockcipher call plus a small and constant number of elementary machine operations. Our constructions work by associating to the i^{th} coordinate of $\mathbb{I}$ a "small" element $\alpha_i \in \mathbb{F}_{2^n}^*$ and multiplying by α_i when one increments that component of the tweak. We illustrate the use of this approach by refining the authenticated-encryption scheme OCB and the message authentication code PMAC, yielding variants of these algorithms, OCB1 and PMAC1, that are simpler and faster than the original schemes, and yet have simpler proofs. Our results bolster the thesis of Liskov, Rivest, and Wagner [12] that a desirable approach for designing modes of operation is to start from a tweakable blockcipher. We elaborate on their idea, suggesting the kind of tweak space, usage-discipline, and blockcipher-based instantiations that give rise to simple and efficient modes of operation of a conventional blockciphers.

Key words: authenticated encryption, modes of operation, OCB, PMAC, provable security, tweakable blockciphers.

1 Introduction

Liskov, Rivest and Wagner [12] defined the notion of a tweakable blockcipher and put forward the thesis that these objects make a good starting point for doing blockcipher-based cryptographic design. Here we describe an elegant way to build a tweakable blockcipher $\widetilde{E}$ out of an ordinary blockcipher E. Used as intended, our constructions, XE and XEX, add just a few machine instructions to the cost of computing E. We illustrate the use of these constructions by improving on the authenticated-encryption scheme OCB [16] and the message authentication code PMAC [5]. Our work helps to turn tweakable blockciphers into practical tools—perhaps the tool of choice for designing ambitious modes of operation.

Tweakable blockciphers. Schroeppel [17] designed a blockcipher, Hasty Pudding, wherein the user supplies a non-secret *spice* and changing any bit of this spice produces a completely different permutation. There is no significant computational cost associated to changing the spice. Liskov, Rivest, and Wagner [12] went on to formally define the syntax and security measures for such a *tweakable* blockcipher, and they suggested that this abstraction makes a desirable starting point to design modes of operation and prove them secure. They suggested some ways to build a tweakable

blockcipher $\widetilde{E}$ out of a standard blockcipher E, as well as ways to modify existing blockcipher designs to incorporate a tweak. They illustrated the use of these objects. Formally, a tweakable blockcipher is a map $\widetilde{E}\colon \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ where each $\widetilde{E}_K^T(\cdot) = \widetilde{E}(K, T, \cdot)$ is a permutation and $\mathcal{T}$ is the set of *tweaks*.

Our contributions. We suggest highly efficient ways to make a tweakable blockcipher $\widetilde{E}$ out of an ordinary blockcipher E. (See Appendix A for the best constructions formerly known.) Our *powering-up* constructions, XE and XEX, assume that the tweak space is of the form $\{0,1\}^n \times \mathbb{I}$ where n is the blocksize of E (say $n = 128$) and $\mathbb{I}$ is a set of tuples of integers, such as $\mathbb{I} = [1\,..\,2^{64}] \times [0\,..\,10]$. Our XE construction turns a cpa-secure blockcipher E into a cpa-secure tweakable blockcipher $\widetilde{E}$, while XEX turns a cca-secure blockcipher E into a cca-secure tweakable blockcipher $\widetilde{E}$ (cpa stands for chosen-plaintext attack and cca for chosen-ciphertext attack). The constructions are efficient to the extent that tweaks arise in sequence, with most tweaks $(N, \mathbf{i})$ being identical to the prior tweak $(N, \mathbf{i}')$ except for incrementing some component of $\mathbf{i}$.

As an illustrative and useful example, consider making a tweakable blockcipher $\widetilde{E}\colon \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ out of a conventional blockcipher $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ by defining $\widetilde{E}_K^{N\,i\,j}(M) = E_K(M \oplus \Delta) \oplus \Delta$ where $\Delta = 2^i 3^j \mathcal{N}$ and $\mathcal{N} = E_K(N)$. Arithmetic is done in the finite field $\mathbb{F}_{2^n}$. For concreteness, assume $n = 128$. Then we show that $\widetilde{E}$ is a secure tweakable blockcipher for tweak space $\mathcal{T} = \{0,1\}^n \times [1\,..\,2^{64}] \times [0\,..\,10]$, say. Secure means that $\widetilde{E}$ is good as a strong, tweakable PRP as long as E is secure as a strong, untweakable PRP. Verifying the correctness of the construction involves computing discrete logs in $\mathbb{F}_{2^n}$. Computing $\widetilde{E}_K^{N\,i\,j}(X)$ will usually cost about 1 shift, 1 conditional, and 3–4 xors more than computing an $E_K(X)$ value.

We illustrate how the use of tweakable blockciphers during mode design, followed by the instantiation of the tweakable blockcipher with an ordinary blockcipher by one of our construction, can give rise to modes that are not only simpler, easier to design, and easier prove correct, but also faster. We do this by refining two highly-optimized modes, OCB [16] and PMAC [5], yielding two new modes, OCB1 and PMAC1. The new modes are easier to understand, easier to implement, and faster than the old ones. OCB1 eliminates the utility of preprocessing, saving a blockcipher call. Computing "offsets" in the new modes does not involve Gray-code sequence or counting the number of trailing zero bits in successive integers; all the relevant operations are constant time and independent of the exact index of a message block.

Intuition behind the powering-up construction. In a nutshell, the idea for our construction can be explained like this. Apart from Gray-code reordering, PMAC authenticated an m-block message using a sequence of offsets $L, 2L, 3L, \ldots, (m-1)L$, where multiplication is in the finite field $\mathbb{F}_{2^n}$ and where $L = E_K(0^n)$ is a variant of the underlying key K. When a "special" offset was needed in PMAC we a value $\mathit{huge} \cdot L$, was added (xored) into the current offset, where *huge* is so large a number that it would never, in practice, be among the above-mentioned multiples of L. What we now do instead is to use the easier-to-compute sequence of offsets $2^1 L, 2^2 L, \ldots, 2^{m-1} L$. We insist that our field be represented using a primitive polynomial instead of merely an irreducible one, which ensures that $2^1, 2^2, 2^3, \ldots, 2^{2^n-1}$ will all be distinct. When a "special" offset is needed we can no longer add to the current offset some huge constant times L and expect this never to land on a point in $2^1 L, 2^2 L, \ldots, 2^{m-1} L$. Instead, we just multiply the current offset by 3, say, instead of 2. If the index of 3 (in $\mathbb{F}_{2^n}^*$) is enormous relative to the base 2, yet still much less than than maximal possible value of $2^n - 1$,then multiplying by 3 is equivalent to multiplying by 2^{huge} and $2^i 3\ L$ won't be among of $2^1 L, 2^2 L, \ldots, 2^{m-1} L$ for any reasonable value of m. The current paper will make all of the ideas of this paragraph precise.

2 Preliminaries

THE FIELD WITH 2^n POINTS. Let $\mathbb{F}_{2^n}$ denote the field with 2^n points and let $\mathbb{F}_{2^n}^*$ denote the multiplicative subgroup of this field (which contains $2^n - 1$ points). We interchangeably think of a point a in $\mathbb{F}_{2^n}$ in any of the following ways: (i) as an abstract point in the field; (ii) as an n-bit string $a_{n-1} \dots a_1 a_0 \in \{0,1\}^n$; (iii) as a formal polynomial $a(\mathtt{x}) = a_{n-1}\mathtt{x}^{n-1} + \cdots + a_1\mathtt{x} + a_0$ with binary coefficients; (iv) as an integer between 0 and $2^n - 1$, where the string $a \in \{0,1\}^n$ corresponds to the number $\sum_{i=0}^{n-1} a_i 2^i$. For example, one can regard the string $a = 0^{125}101$ as a 128-bit string, as the number 5, as the polynomial $\mathtt{x}^2 + 1$, or as an abstract point in $\mathbb{F}_{2^{128}}$.

To add two points in $\mathbb{F}_{2^n}$, take their bitwise xor. We denote this operation by $a \oplus b$. To multiply two points, we first fix a primitive polynomial p_n having binary coefficients and degree n: say the lexicographically first primitive polynomial among the primitive degree n polynomials having a minimum number of nonzero coefficients. For $n = 128$, the indicated polynomial is $p_{128} = \mathtt{x}^{128} + \mathtt{x}^7 + \mathtt{x}^2 + \mathtt{x} + 1$. For $n = 64$ we have $p_{64} = \mathtt{x}^{64} + \mathtt{x}^4 + \mathtt{x}^3 + \mathtt{x} + 1$. Saying that $p_n(\mathtt{x})$ is primitive means that it is irreducible over $\mathbb{F}_2$ and and $\mathtt{x} = 2$ generates all of $\mathbb{F}_{2^n}^*$ (so the point 2 has order $2^n - 1$). To multiply $a, b \in \mathbb{F}_{2^n}$, which we denote ab, regard a and b as polynomials $a(\mathtt{x}) = a_{n-1}\mathtt{x}^{n-1} + \cdots + a_1\mathtt{x} + a_0$ and $b(\mathtt{x}) = b_{n-1}\mathtt{x}^{n-1} + \cdots + b_1\mathtt{x} + b_0$, form their product $c(\mathtt{x})$ over $\mathbb{F}_2$, and take the remainder one gets when dividing $c(\mathtt{x})$ by $p_n(\mathtt{x})$.

It is computationally simple to multiply $a \in \{0,1\}^n$ by 2 (to "double" a). We illustrate the method for $n = 128$, in which case multiplying $a = a_{n-1} \cdots a_1 a_0$ by $\mathtt{x} = 2$ yields $a_{n-1}\mathtt{x}^n + a_{n-2}\mathtt{x}^{n-1} + a_1\mathtt{x}^2 + a_0\mathtt{x}$. Thus, if the first bit of a is 0, then $2a = a{\ll}1$, the left-shift of a by one bit. If the first bit of a is 1 then we must add $\mathtt{x}^{128}$ to $a{\ll}1$. Since $p_{128} = \mathtt{x}^{128} + \mathtt{x}^7 + \mathtt{x}^2 + \mathtt{x} + 1 = 0$ we know that $\mathtt{x}^{128} = \mathtt{x}^7 + \mathtt{x}^2 + \mathtt{x} + 1$, so adding $\mathtt{x}^{128}$ means to xor by $0^{120}10^41^3$. In summary, when $n = 128$ we have that $2a = a{\ll}1$ if $\mathsf{firstbit}(a) = 0$ and $2a = (a{\ll}1) \oplus 0^{120}10^41^3$ if $\mathsf{firstbit}(a) = 1$.

One can easily multiply by other small constants as well: $3a = 2a \oplus a$ and $5a = 2(2a) \oplus a$ and $7a = 2(2a) \oplus 2a \oplus a$ and so forth.

BLOCKCIPHERS AND TWEAKABLE BLOCKCIPHERS. We review the standard definitions for blockciphers and their security [2] and the extension of these notions to tweakable blockciphers [12].

A *blockcipher* is a function $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ where $n \geq 1$ is a number and $\mathcal{K}$ is a finite nonempty set and $E(K, \cdot) = E_K(\cdot)$ is a permutation for all $K \in \mathcal{K}$. A *tweakable blockcipher* is a function $\widetilde{E}$: $\mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ where $n \geq 1$ is a number and $\mathcal{K}$ is a finite nonempty set and $\mathcal{T}$ is a nonempty set and $\widetilde{E}(K, T, \cdot) = \widetilde{E}_K^T(\cdot)$ is a permutation for all $K \in \mathcal{K}$ and $T \in \mathcal{T}$. For blockciphers and tweakable blockciphers we call n the *blocksize* and we call $\mathcal{K}$ the *key space*. For tweakable blockciphers we call $\mathcal{T}$ the *tweak space*.

Let $\mathrm{Perm}(n)$ be the set of all permutations on n bits. Let $\mathrm{Perm}(\mathcal{T}, n)$ be the set of all mappings from $\mathcal{T}$ to permutations on n bits. In writing $\pi \xleftarrow{\$} \mathrm{Perm}(n)$ we are choosing a random permutation $\pi(\cdot)$ on $\{0,1\}^n$. In writing $\pi \xleftarrow{\$} \mathrm{Perm}(\mathcal{T}, n)$ we are choosing a random permutation $\pi(T, \cdot) = \pi_T(\cdot)$ on $\{0,1\}^n$ for each $T \in \mathcal{T}$. If E: $\mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ is a blockcipher then its inverse is the blockcipher $D = E^{-1}$ where D: $\mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ is defined by $D(K, Y) = D_K(Y)$ being the unique point X such that $E_K(X) = Y$. If $\widetilde{E}$: $\mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ is a tweakable blockcipher then its inverse is the tweakable blockcipher $\widetilde{D} = \widetilde{E}^{-1}$ where $\widetilde{D}$: $\mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ is defined by $\widetilde{D}(K, Y) = \widetilde{D}_K(Y)$ being the unique point X such that $\widetilde{E}_K^T(X) = Y$.

An adversary is a probabilistic algorithm with access to zero or more oracles. Without loss of generality, adversaries never ask a query for which the answer is trivially known (that is, an adversary does not repeat a query, does not ask $D_K(Y)$ after receiving Y in response to a query $E_K(X)$, and so forth). Oracles will have an implicit domain of valid queries and, for convenience,

we assume that all adversarial queries lie within that domain. This is not a significant restriction because membership can be easily tested for all domains of interest to us.

We have the following definitions for security.

Definition 1 [Blockcipher and tweakable-blockcipher security] *Let* $E\colon \mathcal{K}\times\{0,1\}^n \to \{0,1\}^n$ *be a blockcipher, let* $\widetilde{E}\colon \mathcal{K}\times\mathcal{T}\times\{0,1\}^n \to \{0,1\}^n$ *be a tweakable blockcipher, and let* A *be an adversary. Then:*

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{prp}}_{E}(A) &= \Pr[K \xleftarrow{\$} \mathcal{K}:\ A^{E_K(\cdot)} \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \mathrm{Perm}(n):\ A^{\pi(\cdot)} \Rightarrow 1] \\
\mathbf{Adv}^{\pm\mathrm{prp}}_{E}(A) &= \Pr[K \xleftarrow{\$} \mathcal{K}:\ A^{E_K(\cdot)\,D_K(\cdot)} \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \mathrm{Perm}(n):\ A^{\pi(\cdot)\,\pi^{-1}(\cdot)} \Rightarrow 1] \\
\mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\widetilde{E}}(A) &= \Pr[K \xleftarrow{\$} \mathcal{K}:\ A^{\widetilde{E}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \mathrm{Perm}(\mathcal{T},n):\ A^{\pi(\cdot,\cdot)} \Rightarrow 1] \\
\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\widetilde{E}}(A) &= \Pr[K \xleftarrow{\$} \mathcal{K}:\ A^{\widetilde{E}_K(\cdot,\cdot)\,\widetilde{D}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \mathrm{Perm}(\mathcal{T},n):\ A^{\pi(\cdot,\cdot)\,\pi^{-1}(\cdot,\cdot)} \Rightarrow 1] \qquad \square
\end{aligned}$$

Of course D and $\widetilde{D}$ denote the inverses of blockciphers E and $\widetilde{E}$. By $\Pr[\text{experiment}:\ \text{event}]$ we mean the probability of the specified event after performing the specified experiment. In writing $A \Rightarrow 1$ (read "A outputs 1") we are referring to the event that the adversary A outputs the bit 1.

In the usual way we lift advantage measures that depend on an adversary to advantage measures that depend on named resources: $\mathbf{Adv}^{\mathrm{xxx}}_{\Pi}(\mathcal{R}) = \max_A\{\mathbf{Adv}^{\mathrm{xxx}}_{\Pi}(A)\}$ over all adversaries A that use resources at most $\mathcal{R}$. The resources of interest to us are the total number of oracle queries q and the total length of those queries σ and the running time t. For convenience, the total length of queries will be measured in n-bit blocks, for some understood value of n, so a query X contributes $|X|_n$ to the total, where $|X|_n$ means $\max\{|X|/n, 1\}$. Running time, by convention, includes the description size of the algorithm relative to some standard encoding. When we speak of authenticity, the block length of the adversary's output is included in σ

3 The XE and XEX Constructions

Goals. We want to support tweak sets that look like $\mathcal{T} = \{0,1\}^n \times \mathbb{I}$ where $\mathbb{I}$ is a set of tuples of integers. In particular, we want to be able to make $\mathbb{I}$ the cross product of a large subrange of integers, like $[1\,..\,2^{n/2}]$, by the cross product of small ranges of integers, like $[0\,..\,10]\times[0\,..\,10]$. Thus an example tweak space is $\mathcal{T} = \{0,1\}^n \times [1\,..\,2^{n/2}] \times [0\,..\,10] \times [0\,..\,10]$. We will assume that tweaks arise in some sequence $T_1, T_2, \ldots$ and that "most" tweaks are an increment of the immediately prior tweak. When we say that one tweak $T = (N, i_1, \ldots, i_k)$ is the increment of another we mean that one of $i_1, \ldots, i_k$ got incremented and everything else stayed the same. The second component of tweak $(N, i_1, \ldots, i_k)$, meaning i_1, is the component that we expect to get incremented most often. We want there to be a simple, constant-time procedure to increment a tweak at any given component of $\mathbb{I}$. To increment a tweak you shouldn't have to go to memory or consult a table or examine which number tweak you are on. Incrementing tweaks should be endian-independent and avoid extended-precision arithmetic. Efficiently incrementing tweaks shouldn't require precomputation.

Tweaks that are not the increment of a prior tweak will also arise, and they will typically look like $(N, 1, 0\ldots, 0)$. Constructions should be reasonably efficient in dealing with such tweaks, too.

We emphasize that the efficiency measure we are focusing on is not the cost of computing $\widetilde{E}^T_K(X)$ from scratch—by that measure our constructions will not be particularly good. Instead, we are specifically interested in the cost of computing $\widetilde{E}^T_K(X)$ given that one has just computed $\widetilde{E}^S_K(X')$ where T is obtained by incrementing S at some component. Most often that component

will have been the second component of S. This is the situation that most often arises (or that can be made to most often arise) in constructions that use tweakable blockciphers.

TWEAKING WITH $\Delta = 2^i\,\mathcal{N}$. Recall that we have chosen to represent points in $\mathbb{F}_{2^n}$ using a primitive polynomial, not just an irreducible one. This means that the point 2 is a generator of the multiplicative subgroup $\mathbb{F}_{2^n}^*$ of $\mathbb{F}_{2^n}$: the points $1, 2, 2^2, 2^3, \ldots, 2^{2^n-2}$ are all distinct. This property turns out to be the crucial one that lets us construct from a blockcipher $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ a tweakable blockcipher $\widetilde{E}\colon \mathcal{K} \times (\{0,1\}^n \times [1\,..\,2^n-2]) \times \{0,1\}^n \to \{0,1\}^n$ by way of

$$\widetilde{E}_K^{N\,i}(M) \;=\; E_K(M \oplus \Delta) \oplus \Delta \quad \text{where} \quad \Delta = 2^i\,\mathcal{N} \quad \text{and} \quad \mathcal{N} = E_K(N). \tag{1}$$

The tweak set is $\mathcal{T} = \{0,1\}^n \times \mathbb{I}$ where $\mathbb{I} = [1\,..\,2^n-2]$ and the tweakable blockcipher just described is denoted $\widetilde{E} = \mathrm{XEX}[E, 2^{\mathbb{I}}]$. When computing the sequence of values $\widetilde{E}_K^{N\,1}(M_1), \ldots, \widetilde{E}_K^{N\,m-1}(M_{m-1})$ each $\widetilde{E}_K^{N\,i}(M_i)$ computation but the first uses one blockcipher call and one doubling operation. Doubling takes a shift followed by a conditional xor. We call the construction above, and all the subsequent constructions of this section, *powering-up* constructions.

TWEAKING BY $\Delta = 2^i 3^j\,\mathcal{N}$. We may want tweaks that look like (N, i, j) where $N \in \{0,1\}^n$ and where i is an integer from a large set of integers $\mathbb{I}$, like $\mathbb{I} = [1\,..\,2^{n/2}]$, and j is an integer from some small set of integers $\mathbb{J}$, like $\mathbb{J} = \{0,1\}$ or $\mathbb{J} = [0\,..\,10]$. To get the "diversity" associated to the various j-values we just multiply by 3 instead of 2. That is, we construct from a blockcipher $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ a tweakable blockcipher $\widetilde{E}\colon \mathcal{K} \times (\{0,1\}^n \times \mathbb{I} \times \mathbb{J}) \times \{0,1\}^n \to \{0,1\}^n$ by way of

$$\widetilde{E}_K^{N\,i\,j}(M) \;=\; E_K(M \oplus \Delta) \oplus \Delta \quad \text{where} \quad \Delta = 2^i 3^j\,\mathcal{N} \quad \text{and} \quad \mathcal{N} = E_K(N). \tag{2}$$

The tweakable blockcipher just described is denoted $\widetilde{E} = \mathrm{XEX}[E, 2^{\mathbb{I}}3^{\mathbb{J}}]$. Incrementing the tweak at component i is done by doubling, while incrementing the tweak at component j is done by tripling.

THE GENERAL XEX CONSTRUCTION. In greater generally, we fix a list of bases $\alpha_1, \alpha_2, \ldots, \alpha_k$. Each base is an element of $\mathbb{F}_{2^n}^*$. The construction is

$$\widetilde{E}_K^{N\,i_1\,i_2\,\cdots\,i_k}(M) \;=\; E_K(M \oplus \Delta) \oplus \Delta \quad \text{where} \quad \Delta = \alpha_1^{i_1}\alpha_2^{i_2}\cdots\alpha_k^{i_k}\,\mathcal{N} \quad \text{and} \quad \mathcal{N} = E_K(N). \tag{3}$$

Each i_ℓ has a domain of allowed values $\mathbb{I}_\ell \subseteq \mathbb{Z}$ so the tweak set is $\mathcal{T} = \{0,1\}^n \times \mathbb{I}_1 \times \cdots \times \mathbb{I}_k$. See the right-hand side of Figure 1.

Definition 2 [XEX construction] *Let $E\colon \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher, let $\alpha_1, \ldots, \alpha_k \in \mathbb{F}_{2^n}^*$ be group elements, and let $\mathbb{I}_1, \ldots, \mathbb{I}_k \subseteq \mathbb{Z}$. Then $\widetilde{E} = \mathsf{XEX}[E, \alpha_1^{\mathbb{I}_1} \cdots \alpha_k^{\mathbb{I}_k}]$ is the tweakable blockcipher $\widetilde{E}\colon \mathcal{K} \times (\{0,1\}^n \times \mathbb{I}_1 \cdots \mathbb{I}_k) \times \{0,1\}^n \to \{0,1\}^n$ defined by $\widetilde{E}_K^{N i_1 \ldots i_k}(M) = E_K(M \oplus \Delta) \oplus \Delta$ where $\Delta = \alpha_1^{i_1}\alpha_2^{i_2}\cdots\alpha_k^{i_k}\,\mathcal{N}$ and $\mathcal{N} = E_K(N)$.*

THE XE CONSTRUCTION. As made clear in the work of Liskov, Rivest, and Wagner [12], constructions of the form $\widetilde{E}_K^T(M) = E_K(M \oplus \Delta) \oplus \Delta$ aim for chosen-ciphertext attack (cca) security while for chosen-plaintext attack (cpa) security you can omit the outer offset, setting $\widetilde{E}_K^T(M) = E_K(M \oplus \Delta)$. Thus we consider the construction

$$\widetilde{E}_K^{N\,i_1\,i_2\,\cdots\,i_k}(M) \;=\; E_K(M \oplus \Delta) \quad \text{where} \quad \Delta = \alpha_1^{i_1}\alpha_2^{i_2}\cdots\alpha_k^{i_k}\,\mathcal{N} \quad \text{and} \quad \mathcal{N} = E_K(N). \tag{4}$$

The tweak space is $\mathcal{T} = \{0,1\}^n \times \mathbb{I}_1 \times \cdots \times \mathbb{I}_k$ and we denote this construction $\widetilde{E} = \mathrm{XE}[E, \alpha_i^{\mathbb{I}_1} \cdots \alpha_i^{\mathbb{I}_k}]$. It is slightly more efficient than XEX, saving one xor. See the left-hand side of Figure 1.

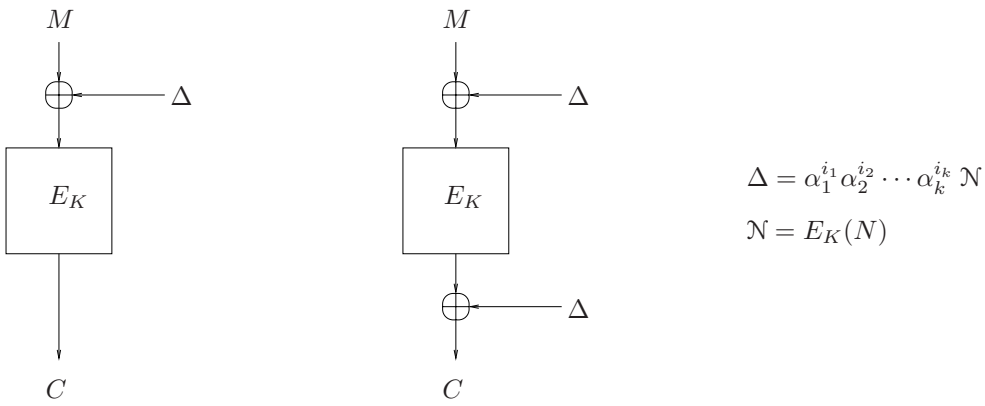


Figure 1: The XE and XEX constructions. Starting with a blockcipher $E\colon \mathcal{K}\times\{0,1\}^n \to \{0,1\}^n$ and points $\alpha_1,\ldots,\alpha_k \in \mathbb{F}^*_{2^n}$ and sets $\mathbb{I}_1,\ldots,\mathbb{I}_k \subseteq \mathbb{Z}$ we construct the blockcipher $\widetilde{E} = \mathrm{XE}[E, \alpha_1^{\mathbb{I}_1}\cdots\alpha_k^{\mathbb{I}_k}]$ (shown on the left) and $\widetilde{\boldsymbol{E}} = \mathrm{XEX}[E, \alpha_1^{\mathbb{I}_1}\cdots\alpha_k^{\mathbb{I}_k}]$ (shown on the right). Multiplication (to form Δ) is in $\mathbb{F}_{2^n}$.

Definition 3 [XE construction] *Let $E\colon \mathcal{K}\times\mathcal{T}\times\{0,1\}^n \to \{0,1\}^n$ be a blockcipher and let $\alpha_1,\ldots,\alpha_k \in \mathbb{F}^*_{2^n}$ be group elements and let $\mathbb{I}_1,\ldots,\mathbb{I}_k \subseteq \mathbb{Z}$. Then $\widetilde{E} = \mathsf{XE}[E, \alpha_1^{\mathbb{I}_1}\cdots\alpha_k^{\mathbb{I}_k}]$ is the tweakable blockcipher $\widetilde{E}\colon \mathcal{K}\times(\{0,1\}^n\times\mathbb{I}_1\cdots\mathbb{I}_k)\times\{0,1\}^n \to \{0,1\}^n$ defined by $\widetilde{E}_K^{Ni_1\ldots i_k}(M) = E_K(M\oplus\Delta)$ where $\Delta = \alpha_1^{i_1}\alpha_2^{i_2}\cdots\alpha_k^{i_k}\,\mathcal{N}$ and $\mathcal{N} = E_K(N)$.*

4 Parameter Sets Yielding Unique Representations

We will soon see that our XE and XEX constructions only "work" when the $\alpha_1^{i_1}\cdots\alpha_k^{i_k}$-values are distinct over the allowed indices $(i_1,\ldots,i_k) \in \mathbb{I}_1\times\cdots\times\mathbb{I}_k$. This motivates the following definition:

Definition 4 [Unique representations] *Fix a group G. A **choice of parameters** is a list $\alpha_1,\ldots,\alpha_k \in G$ of **bases** and a set $\mathbb{I}_1\times\cdots\times\mathbb{I}_k \subseteq \mathbb{Z}^k$ of **allowed indices**,. We say that the choice of parameters provides **unique representations** if for every $(i_1,\ldots,i_k)$, $(j_1,\ldots,j_k) \in \mathbb{I}_1\times\cdots\times\mathbb{I}_k$ we have that $\alpha_1^{i_1}\cdots\alpha_k^{i_k} = \alpha_1^{j_1}\cdots\alpha_k^{j_k}$ implies $(i_1,\ldots,i_k) = (j_1,\ldots,j_k)$, and, in addition, $\alpha_1^{i_1}\cdots\alpha_k^{i_k} = 1$ for no $(i_1,\ldots,i_k) \in \mathbb{I}_1\times\cdots\times\mathbb{I}_k$ other than $(0,\ldots,0)$.* □

In other words, representable points are uniquely representable: any group element $\alpha_1^{i_1}\cdots\alpha_k^{i_k}$ that you can represent using allowed indices $i_1,\ldots,i_k$ you can represent in no *other* way using allowed indices. The second condition adds that if 1 can be represented at all then its representation must be the canonical one, $(0,\ldots,0)$. This second condition is vacuous when $(0,\ldots,0) \in \mathbb{I}_1\times\cdots\times\mathbb{I}_k$.

For tweak spaces of practical interest some computer-aided computation can be used to help choose and verify that a given choice of parameters provides unique representations. The following result gives some examples for $\mathbb{F}^*_{2^{128}}$. The technique for verifying unique representations involves computing discrete logs within the group.

Proposition 5 [Can use bases 2, 3, 7 when $n = 128$] *In the group $\mathbb{F}^*_{2^{128}}$ the following choices for parameters provide unique representations:*

(1) *Base $\alpha_1 = 2$ with allowed indices $[-2^{126}\,..\,2^{126}]$.*

(2) *Bases $\alpha_1,\alpha_2 = 2,3$ with allowed indices $[-2^{115}\,..\,2^{115}]\times[-2^{10}\,..\,2^{10}]$.*

(3) *Bases $\alpha_1,\alpha_2,\alpha_3 = 2,3,7$ with allowed indices $[-2^{108}\,..\,2^{108}]\times[-2^{7}\,..\,2^{7}]\times[-2^{7}\,..\,2^{7}]$.* □

We comment that the above result does depend on the choice of representations for the group; recall that we fixed a representation of $\mathbb{F}_{2^n}^*$ using the lexicographically first primitive polynomial.

Proof: For statement (1) recall that 2 is a generator of the group (by our choice or irreducible polynomial) and the order of the group $\mathbb{F}_{2^{128}}^*$ is $2^{128}-1$ and so $2^i = 2^j$ iff $i = j \pmod{2^{128}-1}$ and so any contiguous range of $2^{128}-1$ or fewer integers will provide unique representations with respect to base 2.

To prove statement (2) we need to compute $\log_2 3$ in the group $\mathbb{F}_{2^{128}}^*$:

$$\log_2 3 \;=\; 338793687469689340204974836150077311399 \quad \text{(decimal)} \tag{5}$$

This and subsequent discrete logs were computed using a Maple-implementation combining the Pohlig-Hellman [13] and Pollard-rho [14] algorithms. (A naive implementation computes discrete logs in $\mathbb{F}_{2^{128}}^*$ in a few hours.) Now note that $2^a 3^b = 2^{a'} 3^{b'}$ iff $2^a 2^{b \log_2 3} = 2^{a'} 2^{b' \log_2 3}$ iff $2^{a+b\log_2 3} = 2^{a'+b'\log_2 3}$ iff $a + b\log_2 3 = a' + b'\log_2 3 \pmod{2^{128}-1}$ because 2 is a generator of the group $\mathbb{F}_{2^{128}}^*$. Thus $2^a 3^b = 2^{a'} 3^{b'}$ iff $a - a' = (b'-b)\log_2 3 \pmod{2^{128}-1}$. If $b, b' \in [-2^{10} .. 2^{10}]$ then $\Delta_b = b' - b \in [-2^{11} .. 2^{11}]$ and computer-assisted calculation then shows that the smallest value of $\Delta_b \log_2 3 \pmod{2^{128}-1}$ for $\Delta_b \in [-2^{11} .. 2^{11}]$ and $\Delta_b \neq 0$ is $1600 \log_2 3 = 00113a0ce508326c006763c0b80c59f9$ (in hexadecimal) which is about $2^{116.1}$. (By "smallest" we refer to the distance from 0, modulo $2^{128}-1$, so 2^{100} and $(2^{128}-1) - 2^{100}$ are equally small, for example.) Thus if a, a' are restricted to $[-2^{115} .. 2^{115}]$ and b, b' are restricted to $[-2^{10} .. 2^{10}]$ then $\Delta_a = a - a' \leq 2^{116}$ can never equal $\Delta_b \log_2 3 \pmod{2^{128}-1} > 2^{116}$ unless $\Delta_b = 0$. This means that the only solution to $2^a 3^b = 2^{a'} 3^{b'}$ within the specified range is $a = a'$ and $b = b'$.

To prove statement (3) is similar. First we need the value

$$\log_2 7 \;=\; 305046802472688182329780655685899195396 \quad \text{(decimal)} \tag{6}$$

Now $2^a 3^b 7^c = 2^{a'} 3^{b'} 7^{c'}$ iff $a - a' = (b'-b)\log_2 3 + (c'-c)\log_2 7 \pmod{2^{128}-1}$. The smallest value for $\Delta_b \log_2 3 + \Delta_c \log_2 7 \pmod{2^{128}-1}$ when $\Delta_b, \Delta_c \in [-2^8 .. 2^8]$ and at least one of the values is non-zero is $-48\log_2 3 + 31 \log_2 7 \pmod{2^{128}-1} = 00003bfabac91e02b278b7e69a379d18$ (hexadecimal) which is about $2^{109.9}$. So restricting the index for base-2 to $[-2^{108} .. 2^{108}]$ ensures that $a - a' \leq 2^{109}$ while $(b'-b)\log_2 3 + (c'-c)\log_2 7 > 2^{109}$ unless $b = b'$ and $c = c'$ and $a = a'$. ∎

We emphasize that not just any list of bases will work. Notice, for example, that $3^2 = 5$ in $\mathbb{F}_{2^n}^*$ (because $3^2 = (\mathtt{x}+1)^2 = \mathtt{x}^2 + 1 = 5$) so the list of bases $2, 3, 5$ certainly does *not* give unique representations, even for a tiny list of allowed indices like $\mathbb{I}_1 \times \mathbb{I}_2 \times \mathbb{I}_3 = \{0,1,2\} \times \{0,1,2\} \times \{0,1,2\}$.

Similar calculations can be done in other groups; here we state the analogous result for $\mathbb{F}_{2^{64}}^*$.

Proposition 6 [Can use bases 2, 3, 11 when $n = 64$] *In the group $\mathbb{F}_{2^{64}}^*$ the following choices for parameters provide unique representations:*

(1) *Base $\alpha_1 = 2$ with allowed indices $[-2^{62} .. 2^{62}]$.*

(2) *Bases $\alpha_1, \alpha_2 = 2, 3$ with allowed indices $[-2^{51} .. 2^{51}] \times [-2^{10} .. 2^{10}]$.*

(3) *Bases $\alpha_1, \alpha_2, \alpha_3 = 2, 3, 11$ with allowed indices $[-2^{44} .. 2^{44}] \times [-2^7 .. 2^7] \times [-2^7 .. 2^7]$.* □

This time 2, 3, 7 does *not* work as a list of bases (even with a small set of allowed indices, like $[1 .. 64] \times [0 .. 1] \times [0 .. 1]$) due to the fact that $2^{64} = 3^2 \cdot 7$ in this group. Machine-assisted verification is certainly essential here; a relation like that just given is found immediately when computing the possible values for $\Delta_b \log_2 3 + \Delta_c \log_2 7 \pmod{2^{64}-1}$ but might not otherwise be anticipated.

5 Security of XE

The following result proves the security of the XE construction. Later we will prove a more general result, one that encompass the security not only of XE and XEX, but also modes that simultaneously use both constructions and a single underlying key.

Theorem 7 [Security of XE] *Fix $n \geq 1$ and let $\alpha_1, \ldots, \alpha_k \in \mathbb{F}_{2^n}^*$ be base elements and let $\mathbb{I}_1 \times \cdots \times \mathbb{I}_k$ be allowed indices such that these parameters provide unique representations. Fix a blockcipher $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ and let $\widetilde{E} = \mathsf{XE}[E, \alpha_1^{\mathbb{I}_1} \cdots \alpha_k^{\mathbb{I}_k}]$ Then*

$$\mathbf{Adv}^{\widetilde{\text{prp}}}_{\widetilde{E}}(t,q) \quad \leq \quad \mathbf{Adv}^{\text{prp}}_{E}(t',2q) + \frac{4.5\,q^2}{2^n}$$

where $t' = t + ckn(q+1)$ for some absolute constant c. □

In English, the XE construction promotes a cpa-secure blockcipher to a cpa-secure tweakable blockcipher, assuming you have chosen base elements and a range of allowed indices that provides unique representations. The proof of Theorem 7 is omitted due to page limitations.

We could go on to give a similar bound for the security of XEX, but here some added care is needed. Suppose, to be concrete, that we are looking at $\mathsf{XEX}[E, 2^{\mathbb{I}}]$ and $\mathbb{I} = [0\,..\,2^{n-2}]$. Let the adversary ask a deciphering query with ciphertext $C = 0^n$ and tweak $(0^n, 0)$. If the adversary has a valid enciphering oracle then it will get a response of $M = \widetilde{D}_K^{0^n\,0}(0^n) = \mathcal{N}$ where $\mathcal{N} = E_K(0^n)$. This allows the adversary to defeat the cca-security. For example, enciphering $2\mathcal{N}$ with a tweak of $(0^n, 1)$ and enciphering $4\mathcal{N}$ with a tweak of $(0^n, 2)$ will give identical results if the adversary has a valid enciphering oracle. This issue was pointed out by David Wagner. Correspond to this attack, the proof that covers XEX will exclude any tweak of the form $(N, 0, \ldots, 0)$ and we forbade any point other than $(0, \ldots, 0)$ to serve as a representative for 1.

6 Intermixing CPA-Secure and CCA-Secure Building Blocks

A cca-secure tweakable-blockcipher construction will typically be less efficient than a construction that is only cpa-secure. As a result, a minimalist design might use a cpa-secure construction in part and a cca-secure construction elsewhere. Even if one shuns this idea in the interest of economy, preferring to use a cca-secure design throughout, still one may wish to combine two modes of operation—say a MAC based on a cpa-secure tweakable blockcipher and an encryption scheme based on a cca-secure one—so that the final construction will still, in the end, employ both kinds of primitives. This section explains how to define and think about security when one wants to combine cpa-secure and cca-secure building blocks.

First we need to ask why there is any issue here. Can you not take a cpa-secure $\widetilde{E}$ and a cca-secure $\widetilde{\boldsymbol{E}}$, separately key them, and prove security of the construction that uses $\widetilde{E}$ and $\widetilde{\boldsymbol{E}}$ under the assumption that each tweakable blockcipher meets its own definition of security? You can certainly do this. But the problem is that we do not want to insist on two different keys. If using a single key, the two construction may interact badly. Indeed for XE and XEX they *will* interact badly, the two mechanisms acting to jointly defeat each other. To overcome this we will make a small restriction on the combined mechanism: *a tweak T^* may be used for the cpa-secure construction or for the cca-secure construction, but not for both.*

To formalize this we need an enlarged notion of security. To set things up, let us assume that we have recast two constructions into a single one as follows: we have a tweakable blockcipher $E\colon \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ where $\mathcal{T} = \{0,1\} \times \mathcal{T}^*$. We intend that the first component of the

tweak $T \in \mathcal{T}$ indicates if we want the cpa-secure mechanism (this when the first component of the tweak is 0) or if we want the cca-secure mechanism (when the first component of the tweak is 1). We say that such a tweakable blockcipher is *cpa/cca-tagged*. If we combine the XE and XEX constructions using this convention we get the following tweakable blockcipher, XE$^{\text{or}}$XEX.

Definition 8 [XE$^{\text{or}}$XEX construction] *Let $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher, let $\alpha_1, \ldots, \alpha_k \in \mathbb{F}^*_{2^n}$, and let $\mathbb{I}_1, \ldots, \mathbb{I}_k \subseteq \mathbb{Z}$. Then $\widetilde{E} = XE^{or}XEX[E, \alpha_1^{\mathbb{I}_1} \cdots \alpha_k^{\mathbb{I}_k}]$ is the tweakable blockcipher $\widetilde{E}\colon \mathcal{K} \times (\{0,1\} \times \{0,1\}^n \times \mathbb{I}_1 \cdots \mathbb{I}_k) \times \{0,1\}^n \to \{0,1\}^n$ defined by $\widetilde{E}_K^{0\,N\,i_1 \ldots i_k}(M) = E_K(M \oplus \Delta)$ and $\widetilde{E}_K^{1\,N\,i_1 \ldots i_k}(M) = E_K(M \oplus \Delta) \oplus \Delta$ where $\Delta = \alpha_1^{i_1}\alpha_2^{i_2} \cdots \alpha_k^{i_k}\,\mathcal{N}$ and $\mathcal{N} = E_K(N)$.* □

Now we describe how to measure the security of a cpa/cca-tagged tweakable blockcipher. An adversary A launching an attack on such an object will be given two oracles, $A^{e(\cdot,\cdot)\,d(\cdot,\cdot)}$, where the second oracle computes the inverse of the first (meaning $d(T,Y)$ is the unique X such that $e(T,X) = Y$). The adversary is required to be *tag-respecting*, meaning:

- The adversary may not make any query $d(T,Y)$ where the first component of T is 0.
- If the adversary makes an oracle query with a tweak (b,T^*) then it may make no subsequent query with a tweak $(1-b,T^*)$.

The first condition says that the adversary is only allowed "forward" queries if it chooses to use the putative cpa-secure construction for a given tweak T^* (it may use "forward" or "backward" queries if it chooses the putative cca-secure construction for T^*). The second condition says that, for a given tweak (b,T^*), the adversary must commit (at the time of the first query employing T^*) if it wants to use for T^* the putative cpa-secure construction or the putative cca-secure construction. As always, we insist that there be no pointless queries: the adversary may not repeat an $e(T,X)$ query or a $d(T,Y)$ query, and it may not ask $d(T,Y)$ after having learned $Y = e(T,X)$, or ask $e(T,X)$ after having learned $X = d(T,Y)$. The definition for security is now as follows:

Definition 9 [Security of a cpa/cca-tagged blockcipher] *Let $\widetilde{E}\colon \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ be a tweakable blockcipher that is cpa/cca-tagged and let A be an tag-respecting adversary. Then*

$$\mathbf{Adv}^{\widetilde{[\pm]\text{prp}}}_{\widetilde{E}}(A) \;=\; \Pr[K \xleftarrow{\$} \mathcal{K} :\; A^{\widetilde{E}_K(\cdot,\cdot)\,\widetilde{D}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \mathrm{Perm}(\mathcal{T},n) :\; A^{\pi(\cdot,\cdot)\,\pi^{-1}(\cdot,\cdot)} \Rightarrow 1] \quad \square$$

Naturally $\widetilde{D}$, above, is the inverse of $\widetilde{E}$. If one prefers, one can think of not placing the restriction on A that it be tag-respecting but give it advantage 0 if it is not.

7 Security of the XE-or-XEX Construction

We now specify the security of the XE$^{\text{or}}$XEX construction. The result encompasses that XE is $\widetilde{\text{prp}}$-secure and XEX is $\pm\widetilde{\text{prp}}$-secure, assuming that that underlying blockcipher is $\pm$prp-secure. But the result encompasses more, facilitating protocol design where one mixes XE-modified blockciphers and XEX-modified blockciphers, both with the same key. The proof is given in Appendix B.

Theorem 10 [Security of XE$^{\text{or}}$XEX] *Fix $n \geq 1$ and let $\alpha_1, \ldots, \alpha_k \in \mathbb{F}^*_{2^n}$ be base elements and let $\mathbb{I}_1 \times \cdots \times \mathbb{I}_k$ be allowed indices such that these parameters provide unique representations and such that $(0, \ldots, 0) \notin \mathbb{I}_1 \times \cdots \times \mathbb{I}_k$. Fix a blockcipher $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ and let $\widetilde{E} = XE^{or}XEX[E, \alpha_1^{\mathbb{I}_1} \cdots \alpha_k^{\mathbb{I}_k}]$. Then*

$$\mathbf{Adv}^{\widetilde{[\pm]\text{prp}}}_{\widetilde{E}}(t,q) \;\leq\; \mathbf{Adv}^{\pm\text{prp}}_{E}(t', 2q) + \frac{9.5\,q^2}{2^n}$$

where $t' = t + ckn(q+1)$ for some absolute constant c. □

8 The OCB1 Authenticated-Encryption Scheme

We begin by recasting OCB [16] to use a tweakable blockcipher instead of a conventional blockcipher. Liskov, Rivest, and Wagner first did this [12], but our abstraction is different from theirs. First, guided by what we have done so far, we choose a tweak space of $\mathcal{T} = \{0,1\} \times \{0,1\}^n \times [1\,..\,2^{n/2}] \times \{0,1\}$. The first bit of the tweak is the cpa/cca tag and so, conceptually, the tweak space is $\mathcal{T}^* = \{0,1\}^n \times [1\,..\,2^{n/2}] \times \{0,1\}$. Second, we want our tweaks to increase monotonically, so we switch the "special" processing done in OCB (meaning that which involved xoring in $2^{-1}L$) from the penultimate block to the final block. The resulting algorithm is shown in Figure 2 of the appendix.

Algorithm OCB1$[\widetilde{E}, \tau]$ depends on a tweakable blockcipher $\widetilde{E}\colon K \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ and a number $\tau \in [0\,..\,n]$. For clarity, we write

$$\boldsymbol{\pi}_i^N \;\text{ for }\; \widetilde{E}_K^{1\,N\,i\,0} \quad\text{and}\quad \pi_i^N \;\text{ for }\; \widetilde{E}_K^{0\,N\,i\,0} \quad\text{and}\quad \bar{\pi}_i^N \;\text{ for }\; \widetilde{E}_K^{0\,N\,i\,1}$$

letting the K be implicit.

The security of OCB1[Perm$(\mathcal{T}, n)$] is much simpler to prove than the security of OCB[Perm(n)]. To state the result we give a couple of definitions from [16]. For privacy of a nonce-based encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ we use the notion of indistinguishability-from-random-strings, which defines $\mathbf{Adv}^{\mathrm{priv}}_{\Pi}(A)$ as $\Pr[K \xleftarrow{\$} \mathcal{K}\colon\ A^{\mathcal{E}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[A^{\$(\cdot,\cdot)} \Rightarrow 1]$. Here $\$(\cdot,\cdot)$ is an oracle that, on input (N, M), returns $|M|$ random bits. The adversary is not allowed to repeat a nonce N. For authenticity we use the nonce-based notion of integrity of ciphertexts: the adversary is given an encryption oracle $\mathcal{E}_K(\cdot,\cdot)$ and it is said to *forge* if it outputs an $(N, \mathcal{C})$ that is valid and $\mathcal{C}$ was not the result of any prior (N, M) query. The adversary is not allowed to repeat a nonce N while it queries its encryption oracle. We write $\mathbf{Adv}^{\mathrm{auth}}_{\Pi}(A)$ for $\Pr[K \xleftarrow{\$} \mathcal{K}\colon\ A^{\mathcal{E}_K(\cdot,\cdot)}$ forges $]$. We have the following theorem for the information-theoretic security of OCB1.

Theorem 11 [Security of OCB1 with a perfect tweakable blockcipher] *Fix $n \geq 1$ and $\tau \in [0\,..\,n]$. Let $\mathcal{T} = \{0,1\} \times \{0,1\}^n \times [1\,..\,2^{n/2}] \times \{0,1\}$. Let A be an adversary. Then*

$$\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{OCB1}[\mathrm{Perm}(\mathcal{T},n),\tau]}(A) \;=\; 0 \quad \textit{and} \quad \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{OCB1}[\mathrm{Perm}(\mathcal{T},n),\tau]}(A) \;\leq\; \tfrac{2^{n-\tau}}{2^n-1}. \qquad \square$$

The proof of Theorem 11 is given in Appendix C. Note that the authenticity bound is close to $2^{-\tau}$; in particular, $2^{n-\tau}/(2^n-1) \leq 1/(2^\tau-1)$ for all $\tau \geq 2$. The bounds do not degrade with the number of queries asked by the adversary, the length of these queries, or the time the adversary runs.

Corollary 12 [Security of OCB1 with a cpa/cca-tagged tweakable blockcipher] *Fix $n \geq 1$ and $\tau \in [0\,..\,n]$. Let $\mathcal{T} = \{0,1\} \times \{0,1\}^n \times [1\,..\,2^{n/2}] \times \{0,1\}$ and let $\widetilde{E}\colon \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ be a cpa/cca-tagged tweakable blockcipher. Then*

$$\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{OCB1}[\widetilde{E},\tau]}(t,\sigma) \;\leq\; \mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\widetilde{E}}(t',\sigma) \quad \textit{and} \quad \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{OCB1}[\widetilde{E},\tau]}(t,\sigma) \;\leq\; \mathbf{Adv}^{[\pm]\widetilde{\mathrm{prp}}}_{\widetilde{E}}(t',\sigma) + \frac{2^{n-\tau}}{2^n-1}$$

where $t' = t + cn\sigma$ for some absolute constant c. $\square$

The proof is straightforward and is omitted from this draft. Notice that one needs only cpa-security for privacy but authenticity uses our notion that integrates cpa/cca-security. It is here that one has formalized the intuition that the first $m-1$ tweakable-blockcipher calls to OCB1 need to be cca-secure, but the last two calls need only be cpa-secure.

To realize OCB1 with a conventional blockcipher we use the XEorXEX construction. Fix n and assume that $2, 3$ provides unique representations over $[1\,..\,2^{n/2}] \times \{0,1\}$ (as it does for $n = 64$ and $n = 128$). Then we instantiate OCB1 by way of $\widetilde{E} = \mathrm{XE}^{\mathrm{or}}\mathrm{XEX}[E,\ 2^{\mathbb{I}}3^{\mathbb{J}}]$ where $\mathbb{I} = [1\,..\,2^{n/2}]$

and $\mathbb{J} = \{0,1\}$. Overloading the notation, we write write OCB1$[E,\tau]$. The algorithm is shown in Figure 3 of the appendix. Security of the blockcipher-based OCB1 construction follows by combining Theorems 7, 10, and Corollary 12.

Corollary 13 [Security of OCB1 with a blockcipher] *Fix $n \geq 1$ and $\tau \in [0\,..\,n]$. Assume base elements 2, 3 provide unique representations on $[1\,..\,2^{n/2}] \times \{0,1\}$. Let $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher. Then*

- $\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{OCB1}[E,\tau]}(t,\sigma) \;\leq\; \mathbf{Adv}^{\mathrm{prp}}_{E}(t',2\sigma) + \dfrac{4.5\sigma^2}{2^n}$ *and*
- $\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{OCB1}[E,\tau]}(t,\sigma) \;\leq\; \mathbf{Adv}^{\pm\mathrm{prp}}_{E}(t',2\sigma) + \dfrac{9.5\sigma^2}{2^n} + \dfrac{2^{n-\tau}}{2^n-1}$

where $t' = t + cn\sigma$ for some absolute constant c. □

With OCB one was expected to perform preprocessing to compute the value $L = E_K(0^n)$ and to compute a collection of $2^i L$-values. All this is gone in OCB1; preprocessing is not useful for the mode (beyond setting up the underlying blockcipher key). Beyond this, with OCB processing the j^{th} block involved xoring into the current offset a value $L(i) = 2^i L$ where $i = \mathsf{ntz}(j)$ was the number of trailing zero-bits in the index j. In the absence of preprocessing, offset-calculations were thus not constant time. This too is gone.

The previous paragraph notwithstanding, the actual time difference or chip-area difference between an optimized implementation of OCB and OCB1 will be small, since the overhead of OCB over a mode like CBC was already small. The biggest gain is that the mode is simpler to understand, implement, and prove correct, no longer involving things like Gray codes or finding the most efficient way to bring in the right $L(i)$ value at the right time.

9 The PMAC1 Message Authentication Code

As with OCB, we recast PMAC [5] to use a tweakable blockcipher. The resulting algorithm is shown in the appendix as Figure 4. Algorithm PMAC1$[\widetilde{E},\tau]$ depends on a tweakable blockcipher $\widetilde{E}\colon \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ where $\mathcal{T} = \{0,1\}^n \times [1\,..\,2^{n/2}] \times [0\,..\,4]$, and on a number $\tau \in [1\,..\,n]$. Letting K be implicit in the notation, we write

$$\pi_i \;\text{ for }\; \widetilde{E}_K^{0^n\, i\, 2} \quad\text{and}\quad \bar{\pi}_i \;\text{ for }\; \widetilde{E}_K^{0^n\, i\, 3} \quad\text{and}\quad \bar{\bar{\pi}}_i \;\text{ for }\; \widetilde{E}_K^{0^n\, i\, 4}\,.$$

in our description of PMAC1.

We have been intentionally unfaithful in abstracting the original PMAC algorithm. In particular, in PMAC1 the tweak for the final blockcipher call depends on the message length m. This is because, when using the powering-up constructions, it is desirable for tweaks to increase monotonically.

PMAC1 is not only secure as a MAC, but as a pseudorandom function (PRF) from $\{0,1\}^*$ to $\{0,1\}^\tau$. For a PRF with such a signature recall that $\mathbf{Adv}^{\mathrm{prf}}_F(A) = \Pr[K \xleftarrow{\$} \mathcal{K}\colon\; A^{F_K(\cdot)} \Rightarrow 1] - \Pr[\rho \xleftarrow{\$} \mathrm{Rand}(\{0,1\}^*,\tau)\colon\; A^{\rho(\cdot)} \Rightarrow 1]$ where $\mathrm{Rand}(\{0,1\}^*,\tau)$ denotes the set of all functions from $\{0,1\}^*$ to $\{0,1\}^\tau$. The definition is extended in the usual way to give resource-parameterized advantage measures, with σ measuring the total number of n-bit blocks asked in all adversary queries. We have the following theorems. Proofs are omitted for lack of space.

Theorem 14 [Security of PMAC1 with a perfect tweakable blockcipher] *Fix $n \geq 1$ and $\tau \in [1\,..\,\tau]$. Let $\mathcal{T} = \{0,1\}^n \times [1\,..\,2^{n/2}] \times [0\,..\,4]$. Then* $\mathbf{Adv}^{\widetilde{\mathrm{prf}}}_{\mathrm{PMAC1}[\mathrm{Perm}(\mathcal{T},n),\tau]}(\sigma) \leq \sigma^2/2^n$ □

Corollary 15 [Security of PMAC1 with a tweakable blockcipher] *Fix $n \geq 1$ and $\tau \in [1 \,..\, n]$. Let $\mathcal{T} = \{0,1\}^n \times [1 \,..\, 2^{n/2}] \times [0 \,..\, 4]$ and let $\widetilde{E}$: $\mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ be a tweakable blockcipher. Then $\mathbf{Adv}^{\widetilde{\text{prf}}}_{\text{PMAC1}[\widetilde{E},\tau]}(t, \sigma + 1) \leq \mathbf{Adv}^{\widetilde{\text{prp}}}_{\widetilde{E}}(t', \sigma) + \sigma^2/2^n$ where $t' = t + cn\sigma$ for some absolute constant c.* □

To realize PMAC1 starting from a conventional blockcipher we use the XE construction, defining $\widetilde{E} = \text{XE}[E, 2^{\mathbb{I}}3^{\mathbb{J}}]$ with $\mathbb{I} = [1 \,..\, 2^{n/2}]$ and $\mathbb{J} = [0 \,..\, 4]$. Overloading the notation, when E is an n-bit blockcipher we write PMAC1$[E, \tau]$. The algorithm is shown in Figure 5 of the appendix. Security is given below, obtained by combining Corollary 15 and Theorem 7.

Corollary 16 [Security of PMAC1 with a blockcipher] *Fix $n \geq 1$ and $\tau \in [1 \,..\, n]$. Assume base elements $2, 3$ provide unique representations on $[1 \,..\, 2^{n/2}] \times [0 \,..\, 4]$. Let E: $\mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher. Then $\mathbf{Adv}^{\widetilde{\text{prf}}}_{\text{PMAC1}[E,\tau]}(t, \sigma) \leq \mathbf{Adv}^{\text{prp}}_{E}(t', \sigma + 1) + \frac{5.5\,\sigma^2}{2^n}$ where $t' = t + cn\sigma$ for some absolute constant c.* □

With PMAC one had to xor into the current offset a value $L(i) = 2^i L$ where i was the number of trailing zero-bits in the current block index j, so PMAC simplifies offset computations. Writing an optimized PMAC1 implementation is easier than writing an optimized PMAC implementation, as one is freed from worrying about Gray codes, precomputing $L(i)$-values, or finding the most efficient way to bring in the right $L(i)$ value at the right time. PMAC1 enjoys a simpler proof.

10 Discussion

We have gone through a fair amount of effort in order, in the end, to make modest improvements to two little-used modes. This doesn't bother us. We are aiming to find the "right" abstractions and design approaches for devising state-of-the-art modes, and our improvements to OCB and PMAC confirm the conceptual benefit of thinking in terms of tweakable blockciphers. Besides simplifying designs and proofs, our instantiations of tweakable blockciphers help to improve efficiency compared to modes designed directly on top of blockciphers. This is a benefit not formerly envisaged as flowing from the tweakable-blockcipher abstraction. In order to get this efficiency payoff it is necessary that the designer accept certain "design rules". In particular, the tweak space needs to look like $\{0,1\}^n \times \mathbb{BIG} \times \mathbb{SMALL}$ for appropriate sets $\mathbb{BIG}$ and $\mathbb{SMALL}$; one needs to arrange that most tweaks be obtained by incrementing the prior one; and one realizes a tweakable blockcipher by embellishing an ordinary blockcipher using the XE, XEX, or XE$^{\text{or}}$XEX construction.

Tweakable blockciphers open up another possibility hinted at by this paper. When combining two blockcipher-based cryptographic mechanisms (e.g., an encryption scheme and a MAC) it would normally be important to use two different keys. Tweakable blockciphers allow one to use the same underlying key but arrange that the tweaks used by the two mechanisms be kept disjoint. You retain the modularity of design and analysis associated to using separate keys, but you save on key material and key-setup costs. As an example, we have quietly separated the space of tweaks for OCB1 and PMAC1, which lets the two modes be glued together using the construction of [15] without going through a complicated analysis. Definition 9 and Theorem 10 were aimed at combining constructions with one key but disjoint sets of tweaks.

Somewhat strangely, our design has relied on the relative *easiness* of computing discrete logarithms in a modest-size group (you need to know the log base-2 of 3 in $\mathbb{F}_{2^n}$ to be sure that multiplying by 3 is not like multiplying by any "reasonable" power of 2). The authors know of no other example where one needed to compute discrete logs in order to design or verify a blockcipher mode of operation.

References

[1] 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 38th Annual Symposium on Foundations of Computer Science* (FOCS 97), IEEE, 1997.

[2] M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences*, vol. 61, no. 3, Dec 2000. (Full version of paper from *Advances in Cryptology — CRYPTO '94*. Lecture Notes in Computer Science, vol. 839, pp. 340–358, 1994.)

[3] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *Advances in Cryptology — ASIACRYPT '00*. Lecture Notes in Computer Science, T. Okamoto., ed., Springer-Verlag, 2000.

[4] P.Crowley. Mercy: a fast large block cipher for disk sector encryption. *Fast Software Encryption*. Lecture Notes in Computer Science, vol. 1978, Springer-Verlag, pp. 49–63, 2000.

[5] J. Black and P. Rogaway. A block-cipher mode of operation for parallelizable message authentication. *Advances in Cryptology — Eurocrypt '02*. Lecture Notes in Computer Science, vol. 2332, pp. 384–397. Springer-Verlag, 2002.

[6] V. Gligor and P. Donescu. Fast encryption and authentication: XCBC encryption and XECB authentication modes. *Fast Software Encryption*, Lecture Notes in Computer Science, Springer-Verlag, April 2001.

[7] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM,* vol. 33, no. 4, pp. 210–217, 1986.

[8] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, vol. 28, April 1984, pp. 270–299.

[9] S. Halevi and P. Rogaway. A parallelizable enciphering mode. Cryptology ePrint Report 2003/147. July 2003.

[10] J. Kilian and P. Rogaway. How to protect DES against exhaustive key search (an analysis of DESX). *J. of Cryptology*, vol. 14, no. 1, pp. 17–35, 2001.

[11] C. Jutla. Encryption modes with almost free message integrity. *Advances in Cryptology — EUROCRYPT 2001.* Lecture Notes in Computer Science, vol. 2045, B. Pfitzmann, ed., Springer-Verlag, 2001.

[12] M. Liskov, R. Rivest, and D. Wagner. Tweakable block ciphers. *Advances in Cryptology — CRYPTO '02*. Lecture Notes in Computer Science, vol. 2442, pp. 31–46, 2002.)

[13] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Transactions on Information Theory*, vol 24, pp. 106–110, 1978.

[14] J. Pollard. Monte Carlo methods for index computation (mod p). *Mathematics of Computation*, vol. 32, pp. 918–924, 1978.

[15] P. Rogaway. Authenticated-encryption with associated-data. ACM CCS'02. ACM Press, 2002.

[16] P. Rogaway, M. Bellare, and J. Black. OCB: A block-cipher mode of operation for efficient authenticated encryption. TISSEC, August 2003. Earlier version, with T. Krovetz, in *ACM Conference on Computer and Communications Security* (CCS-8), ACM Press, pp. 195–205, 2001.

[17] R. Schroeppel. The hasty pudding cipher. AES candidate submitted to NIST, 1998.

A Tweakable Blockciphers Implicit in Prior Work

The tweakable-blockcipher constructions of Liskov, Rivest, and Wagner [12] include one that always doubles the number of blockcipher calls and one that uses an xor-universal hash-function family. No new or particularly practical suggestions for an xor-universal hash-function family are offered by [12], so we regard the most efficient constructions formerly known to be those implicit in earlier modes [5, 6, 11, 16] but recast in view of Liskov, Rivest, and Wagner [12]. In particular:

- Rogaway, Bellare and Black [16] might be seen as implicitly suggesting making a tweakable blockcipher $\widetilde{E}\colon \mathcal{K} \times (\{0,1\}^n \times [0\,..\,2^{n-2}]) \times \{0,1\}^n \to \{0,1\}^n$ from an ordinary blockcipher $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ by way of $\widetilde{E}_K^{N,i}(X) = E_K(X \oplus \Delta) \oplus \Delta$ where $\Delta = \gamma_i L \oplus R$ and $L = E_K(N)$ and $R = E_K(N \oplus L)$ and γ_i is the i-th Gray-code coefficient. Multiplication is in $\mathbb{F}_{2^n}$.
- Black and Rogaway [5] might be seen as making $\widetilde{E}\colon \mathcal{K} \times [0\,..\,2^{n-2}] \times \{0,1\}^n \to \{0,1\}^n$ out of $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ by $\widetilde{E}_K^i(X) = E_K(X \oplus \Delta)$ where $\Delta = \gamma_i L$ and $L = E_K(0^n)$ and γ_i is as before. Multiplication is in the finite field $\mathbb{F}_{2^n}$.
- The definitions above ignore the "special" treatment afforded to blocks modified by xoring in $2^{-1}L$. The implicit intent [5, 16] was to use this mechanism to enlarge the tweak space by one bit, effectively taking the cross product with $\{0,1\}$ and making a tweak space of $\{0,1\}^n \times [0\,..\,2^{n-2}] \times \{0,1\}$ for OCB and $[0\,..\,2^{n-2}] \times \{0,1\}$ for PMAC.
- Jutla [11] might be seen as suggesting a construction (among others) like $\widetilde{E}\colon (\mathcal{K} \times \mathcal{K}') \times (\{0,1\}^n \times \mathbb{Z}_p^+) \times \{0,1\}^n \to \{0,1\}^n$ by way of $\widetilde{E}_{KK'}^{N,i}(X) = E_K(X \oplus \Delta) \oplus \Delta$ where $\Delta = i\ell \bmod p$ and $\ell = E_{K'}(N)$ and p is the largest prime less than 2^n. Numbers and strings are converted back and forth in the natural way.
- Gligor and Donescu [6] might be seen as suggesting constructions like $\widetilde{E}\colon (\mathcal{K} \times \{0,1\}^n) \times [1\,..\,2^n - 1] \to \{0,1\}^n$ by $\widetilde{E}_{K,r}^i(X) = E_K(X + \delta)$ where $\delta = ir$ and addition is done modulo 2^n.

There are additional ways that one may re-interpret the offset-using modes [5, 6, 11, 16] as suggesting tweakable-blockcipher constructions, but none are as simple and efficient as XE and XEX.

B Proof of Theorem 10 — Security of XEorXEX

Let A be a tag-respecting adversary that attacks $\widetilde{E}$. Say that A runs in time at most t and makes exactly q queries (if it makes fewer oracle queries on a given run then have it ask additional, valid queries until it has made q calls). Without loss of generality assume that A is deterministic.

We start off with a hybrid argument. In particular, we introduce the following hybrids:

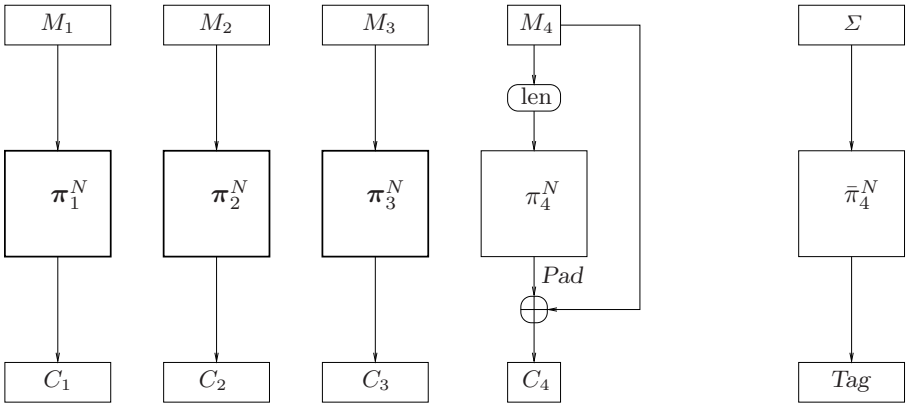
(1) $p_1 = \Pr[K \xleftarrow{\$} \mathcal{K} :\ A^{\widetilde{E}_K(\cdot,\cdot)\,\widetilde{D}_K(\cdot,\cdot)} \Rightarrow 1]$

(2) $p_2 = \Pr[\pi \xleftarrow{\$} \mathrm{Perm}(n) :\ A^{\widetilde{\pi}(\cdot,\cdot)\,\widetilde{\pi}^{-1}(\cdot,\cdot)} \Rightarrow 1]$

(3) $p_3 = \Pr[A^{\$(\cdot,\cdot)\,\$(\cdot,\cdot)} \Rightarrow 1]$

(4) $p_4 = \Pr[\pi \xleftarrow{\$} \mathrm{Perm}(\mathcal{T}, n) :\ A^{\pi(\cdot,\cdot)\,\pi^{-1}(\cdot,\cdot)} \Rightarrow 1]$

Let us explain the notation above. In the experiment associated to p_1 a query $(b, N, i_1, \ldots, i_k)$, M to the adversary's left oracle is answered with $E_K(M \oplus \Delta)$ if $b = 0$ and with $E_K(M \oplus \Delta) \oplus \Delta$ if $b = 1$, where $\Delta = \alpha_1^{i_1} \cdots \alpha_k^{i_k}\, \mathcal{N}$ and $\mathcal{N} = E_K(N)$. A query to the adversary's right oracle is answered with $D_K(M \oplus \Delta)$ if $b = 0$ and $D_K(M \oplus \Delta) \oplus \Delta$ if $b = 1$. Here K is a random key for the

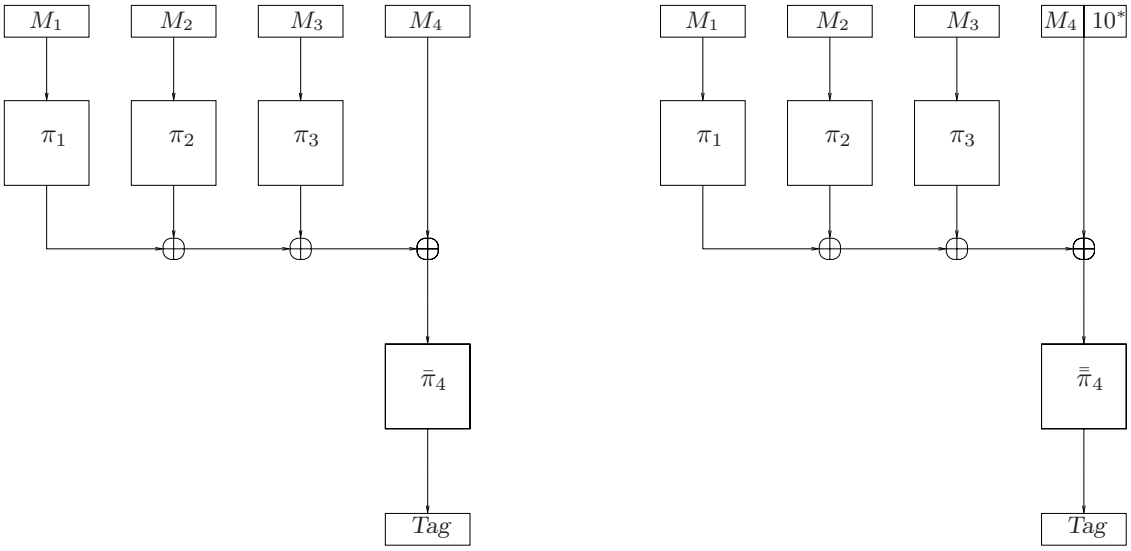


Algorithm $\mathrm{OCB1.Enc}_K^N(M)$	**Algorithm** $\mathrm{OCB1.Dec}_K^N(\mathcal{C})$
Partition M into $M[1]\cdots M[m]$	Partition $\mathcal{C}$ into $C[1]\cdots C[m]\,T$
for $i \in [1\,..\,m-1]$ **do** $C[i] \leftarrow \boldsymbol{\pi}_i^N(M[i])$	**for** $i \in [1\,..\,m-1]$ **do** $M[i] \leftarrow \left(\boldsymbol{\pi}_i^N\right)^{-1}(C[i])$
$Pad \leftarrow \pi_m^N(\mathrm{len}(M[m]))$	$Pad \leftarrow \pi_m^N(\mathrm{len}(C[m]))$
$C[m] \leftarrow M[m] \oplus Pad$	$M[m] \leftarrow C[m] \oplus Pad$
$C \leftarrow C[1]\cdots C[m]$	$M \leftarrow M[1]\cdots M[m]$
$\Sigma \leftarrow M[1] \oplus \cdots \oplus M[m-1] \oplus C[m]0^* \oplus Pad$	$\Sigma \leftarrow M[1] \oplus \cdots \oplus M[m-1] \oplus C[m]0^* \oplus Pad$
$Tag \leftarrow \bar{\pi}_m^N(\Sigma)$	$Tag \leftarrow \pi_m^N(\Sigma)$
$T \leftarrow Tag\,[\text{first } \tau \text{ bits}]$	$T' \leftarrow Tag\,[\text{first } \tau \text{ bits}]$
return $\mathcal{C} \leftarrow C \,\|\, T$	**if** $T = T'$ **then return** M **else return** INVALID

Figure 2: **OCB1**$[\widetilde{E}, \tau]$ using a tweakable blockcipher $\widetilde{E}$ where $\widetilde{E}$: $\mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ and $\mathcal{T} = \{0,1\} \times \{0,1\}^n \times [1\,..\,2^{n/2}] \times \{0,1\}$ and a number $\tau \in [0\,..\,n]$. We write $\boldsymbol{\pi}_i^N$ and π_i^N and $\bar{\pi}_i^N$ for $\widetilde{E}_K^{1\,N\,i\,0}$ and $\widetilde{E}_K^{0\,N\,i\,0}$ and $\widetilde{E}_K^{0\,N\,i\,1}$. The picture on top shows OCB1 encrypting a four-block message $M = M_1M_2M_3M_4$ using nonce N. The resulting ciphertext is $\mathcal{C} = C_1C_2C_3C_4\,Tag$, except that the portion Tag may be truncated. On bottom is a definition of encryption and decryption using key K, nonce N, plaintext M, and ciphertext $\mathcal{C}$.

Algorithm $\mathrm{OCB1.Enc}_K^N(M)$	**Algorithm** $\mathrm{OCB1.Dec}_K^N(\mathcal{C})$
Partition M into $M[1]\cdots M[m]$	Partition $\mathcal{C}$ into $C[1]\cdots C[m]\,T$
$\Delta \leftarrow 2\,E_K(N)$	$\Delta \leftarrow 2\,E_K(N)$
$\Sigma \leftarrow 0^n$	$\Sigma \leftarrow 0^n$
for $i \in [1\,..\,m-1]$ **do**	**for** $i \in [1\,..\,m-1]$ **do**
$\quad C[i] \leftarrow E_K(M[i] \oplus \Delta) \;\oplus\; \Delta$	$\quad M[i] \leftarrow E_K^{-1}(C[i] \oplus \Delta) \;\oplus\; \Delta$
$\quad \Delta \leftarrow 2\Delta$	$\quad \Delta \leftarrow 2\Delta$
$\quad \Sigma \leftarrow \Sigma \oplus M[i]$	$\quad \Sigma \leftarrow \Sigma \oplus M[i]$
$Pad \leftarrow E_K(\mathrm{len}(M[m]) \oplus \Delta)$	$Pad \leftarrow E_K(\mathrm{len}(C[m]) \oplus \Delta)$
$C[m] \leftarrow M[m] \oplus Pad$	$M[m] \leftarrow C[m] \oplus Pad$
$C \leftarrow C[1]\cdots C[m]$	$M \leftarrow M[1]\cdots M[m]$
$\Sigma \leftarrow \Sigma \oplus C[m]0^* \oplus Pad$	$\Sigma \leftarrow \Sigma \oplus C[m]0^* \oplus Pad$
$\Delta \leftarrow 3\Delta$	$\Delta \leftarrow 3\Delta$
$Tag \leftarrow E_K(\Sigma \oplus \Delta)$	$Tag \leftarrow E_K(\Sigma \oplus \Delta)$
$T \leftarrow Tag\,[\text{first } \tau \text{ bits}]$	$T' \leftarrow Tag\,[\text{first } \tau \text{ bits}]$
return $\mathcal{C} \leftarrow C \,\|\, T$	**if** $T = T'$ **then return** M **else return** INVALID

Figure 3: **OCB1**$[E, \tau]$ using a conventional blockcipher E: $\mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ and a number $\tau \in [0\,..\,n]$. This algorithm is the same as $\mathrm{OCB1}[\widetilde{E}, \tau]$ where $\widetilde{E} = \mathrm{XEX}[E, 2^{[1\,..\,2^{n/2}]}3^{\{0,1\}}]$. The key is K, the nonce N, the plaintext is M, and the ciphertext is $\mathcal{C}$.



Algorithm $\mathrm{PMAC1}_K\ (M)$ *//The mode when based on a* ***tweakable*** *blockcipher*

Partition M into $M[1] \cdots M[m]$
for $i \in [1\,..\,m-1]$ **do** $Y[i] \leftarrow \pi_i(M[i])$
if $|M[m]| = n$ **then** $\mathit{Tag} \leftarrow \bar{\pi}_m(Y[1] \oplus \cdots \oplus Y[m-1] \oplus M_m)$
else $\mathit{Tag} \leftarrow \bar{\bar{\pi}}_m(Y[1] \oplus \cdots \oplus Y[m-1] \oplus M_m 10^*)$
$T \leftarrow \mathit{Tag}\,[\text{first } \tau \text{ bits}]$
return T

Figure 4: **PMAC1**$[\widetilde{E}, \tau]$ using a tweakable blockcipher $\widetilde{E}$ and a number τ where $\widetilde{E}$: $\mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ and $\mathcal{T} = \{0,1\}^n \times [1\,..\,2^{n/2}] \times [0\,..\,4]$ and $\tau \in [0\,..\,n]$. We write π_i and $\bar{\pi}_i$ and $\bar{\bar{\pi}}_i$ for $\widetilde{E}_K^{0^n\,i\,2}$ and $\widetilde{E}_K^{0^n\,i\,3}$ and $\widetilde{E}_K^{0^n\,i\,4}$. On top is PMAC1 authenticating $M = M_1M_2M_3M_4$ and yielding a tag Tag, which is truncated (not shown) to give a MAC of T. If $|M_4| = n$ we use the mechanism to the left, otherwise, the mechanism to the right. On bottom is the algorithm showing how to authenticate message M using key K.

Algorithm $\mathrm{PMAC1}_K\ (M)$ *//The mode when based on a* ***conventional*** *blockcipher*

Partition M into $M[1] \cdots M[m]$
$\Theta \leftarrow 10\, E_K(0^n)$
$\Sigma \leftarrow 0^n$
for $i \leftarrow 1$ **to** $m-1$ **do**
  $Y \leftarrow E_K(M[i] \oplus \Theta)$
  $\Sigma \leftarrow \Sigma \oplus Y$
  $\Theta \leftarrow 2\Theta$
if $|M[m]| = n$ **then** $\Theta \leftarrow 3\Theta$, $\Sigma \leftarrow \Sigma \oplus M[m]$
else $\Theta \leftarrow 5\Theta$, $\Sigma \leftarrow \Sigma \oplus M[m]10^*$
$\mathit{Tag} \leftarrow E_K(\Sigma \oplus \Theta)$
$T \leftarrow \mathit{Tag}\,[\text{first } \tau \text{ bits}]$
return T

Figure 5: **PMAC1**$[E, \tau]$ using a conventional blockcipher E: $\mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ and a number $\tau \in [0\,..\,n]$. This algorithm is the same as $\mathrm{PMAC1}[\widetilde{E}, \tau]$ where $\widetilde{E} = \mathrm{XE}[E, 2^{[1\,..\,2^{n/2}]}3^{[0\,..\,4]}]$. The message to authenticate is M and the key is K.

blockcipher E. In the experiment associated to p_2 a query $(b, N, i_1, \ldots, i_k)$, M to the adversary's left oracle is answered with $\pi(M \oplus \Delta)$ if $b = 0$ and $\pi(M \oplus \Delta) \oplus \Delta$ if $b = 1$, where $\Delta = \alpha_1^{i_1} \cdots \alpha_k^{i_k} \mathcal{N}$ and $\mathcal{N} = \pi(N)$. A query to the adversary's right oracle is answered with $D_K(M \oplus \Delta)$ if $b = 0$ and $D_K(M \oplus \Delta) \oplus \Delta$ if $b = 1$. Here π is a random permutation on n bits. In the experiment associated to p_3 a query $(b, N, i_1, \ldots, i_k)$, M to either oracle is answered with n random bits. In the experiment associated to p_4 a left query of $(b, N, i_1, \ldots, i_k)$, M is answered by $\pi_{N,i_1,\ldots,i_k}(M)$ and a right query of $(b, N, i_1, \ldots, i_k)$, M is answered with $\pi^{-1}_{N,i_1,\ldots,i_k}(M)$ where each π_T is a random permutation on n bits.

The value we must bound is $p_1 - p_4 = (p_1 - p_2) + (p_2 - p_3) + (p_3 - p_4)$. Two of these three addends are easy to bound:

- $p_1 - p_2 \le \mathbf{Adv}^{\pm\mathrm{prp}}_E(t', 2q)$ where $t' = t + ckn(q + 1)$ for some constant c depending only on details of the model of computation. This is because one strategy for distinguishing oracles $(F(\cdot), G(\cdot)) = (E_K(\cdot), D_K(\cdot))$ from $(F, G) = (\pi(\cdot), \pi^{-1}(\cdot))$ is embodied by the following adversary $B^{F(\cdot)\,G(\cdot)}$: run $A^{f\,g}$, answering each query $f((b, N, \alpha_1, \ldots, \alpha_k), M)$ by $F(M \oplus \Delta)$ if $b = 0$ and $F(M \oplus \Delta) \oplus \Delta$ if $b = 1$, and answering each query $g((b, N, \alpha_1, \ldots, \alpha_k), M)$ by $G(M \oplus \Delta)$ if $b = 0$ and $G(M \oplus \Delta) \oplus \Delta$ if $b = 1$, where $\Delta = \alpha_1^{i_1} \cdots \alpha_k^{i_k} \mathcal{N}$ and $\mathcal{N} = F(N)$. When A halts with output β have B halt with output β. The time for this adversary is $ckn(q+1)$ beyond the time that the adversary A runs, for some constant c; the total number of queries asked by B is $2q$; and $\mathbf{Adv}^{\pm\mathrm{prp}}_E(B) = p_1 - p_2$.
- $p_3 - p_4 \le 2q^2/2^n$. This is just the standard replacement of a random permutation and its inverse by a pair of random functions. The number of queries is at most $2q$, so the discrepancy between what the permutation gives and what the function gives is at most $0.5 \cdot 2q(2q-1)/2^n \le 2q^2/2^n$. Recall that we have forbidden the types of queries that trivially allow the distinguishing of these two kinds of oracles, such as repeating an oracle query.

We are left with the task of bounding $p_2 - p_3$. This is done with a game-playing argument, as used in works like [10]. We begin with a game, call it Game 2, that accurately simulates the pair of oracles $(\widetilde{\pi}, \widetilde{\pi}^{-1})$ that we have defined. See Figure 6. Also defined in Figure 6 is Game 3, which is obtained by omitting the eight shaded statements.

An inspection of Game 2 will make clear that it provides to the adversary an identical view as that which is obtained by giving the adversary a random $(\widetilde{\pi}(\cdot,\cdot), \widetilde{\pi}^{-1}(\cdot,\cdot))$ oracle. As such,

$$p_2 \;=\; \Pr[\pi \xleftarrow{\$} \mathrm{Perm}(n) :\; A^{\widetilde{\pi}(\cdot,\cdot)\,\widetilde{\pi}^{-1}(\cdot,\cdot)} \Rightarrow 1] \;\;=\;\; \Pr[A^{\text{Game 2}} \Rightarrow 1]\,. \tag{7}$$

On the other hand, it is easy to see that Game 3 simply returns n random bits in response to each query, and so

$$p_3 \;=\; \Pr[A^{\$(\cdot,\cdot)\,\$(\cdot,\cdot)} \Rightarrow 1] \;\;=\;\; \Pr[A^{\text{Game 3}} \Rightarrow 1]\,. \tag{8}$$

The quantity that we must bound is $p_2 - p_3$, and so

$$p_2 - p_3 \;\;=\;\; \Pr[A^{\text{Game 2}} \Rightarrow 1] - \Pr[A^{\text{Game 3}} \Rightarrow 1]\,. \tag{9}$$

Since Games 2 and 3 have been setup to be syntactically identical apart from that which happens following the setting of the flag *bad* to true, the usual game-playing method informs us that

$$p_2 - p_3 \;\;\le\;\; \Pr[A^{\text{Game 3}} \text{ sets } \mathit{bad}\,]\,. \tag{10}$$

To understand Game 3 we will make some modifications to it. Begin by *eliminating* lines 24, 30, 44, and 50. For clarity, we have rewritten Game 3A as Figure 7 This results in a different

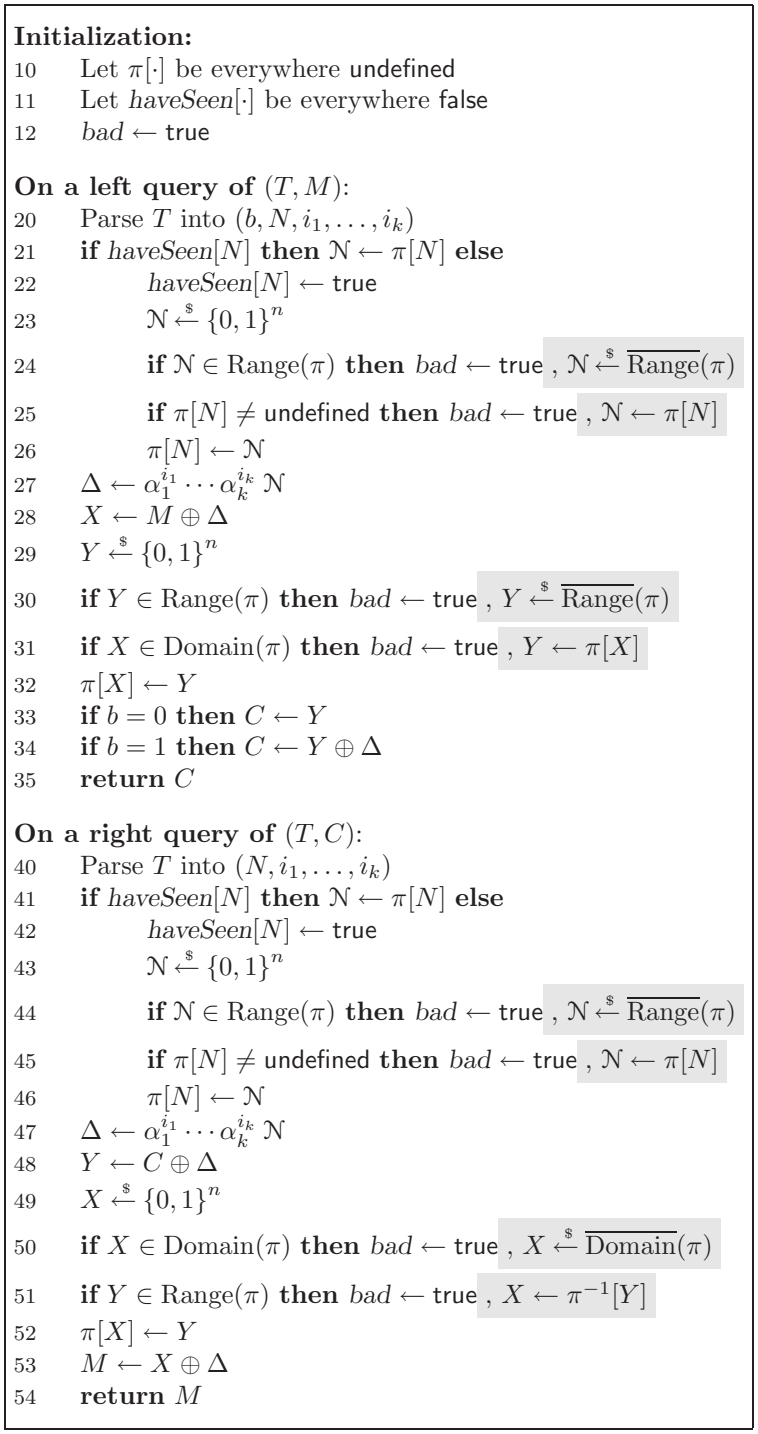


Figure 6: Definition of Game 2, as written, and Game 3, which is obtained by omitting the eight shaded statements.

game, call it Game 3A. In eliminating these lines we may decrease the probability that *bad* gets set to true, but it is easy to see by how much we have lessened this probability: by at most

Initialization:
10 Let $\pi[\cdot]$ be everywhere undefined
11 Let *haveSeen*$[\cdot]$ be everywhere false
12 $bad \leftarrow$ true

On a left query of (T, M):
20 Parse T into $(b, N, i_1, \ldots, i_k)$
21 **if** *haveSeen*$[N]$ **then** $\mathcal{N} \leftarrow \pi[N]$ **else**
22 *haveSeen*$[N] \leftarrow$ true
23 $\mathcal{N} \xleftarrow{\$} \{0,1\}^n$
24 **if** $\pi[N] \neq$ undefined **then** $bad \leftarrow$ true
25 $\pi[N] \leftarrow \mathcal{N}$
26 $\Delta \leftarrow \alpha_1^{i_1} \cdots \alpha_k^{i_k}\, \mathcal{N}$
27 $X \leftarrow M \oplus \Delta$
28 $Y \xleftarrow{\$} \{0,1\}^n$
29 **if** $X \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow$ true
30 $\pi[X] \leftarrow Y$
31 **if** $b = 0$ **then** $C \leftarrow Y$
32 **if** $b = 1$ **then** $C \leftarrow Y \oplus \Delta$
33 **return** C

On a right query of (T, C):
40 Parse T into $(N, i_1, \ldots, i_k)$
41 **if** *haveSeen*$[N]$ **then** $\mathcal{N} \leftarrow \pi[N]$ **else**
42 *haveSeen*$[N] \leftarrow$ true
43 $\mathcal{N} \xleftarrow{\$} \{0,1\}^n$
44 **if** $\pi[N] \neq$ undefined **then** $bad \leftarrow$ true
45 $\pi[N] \leftarrow \mathcal{N}$
46 $\Delta \leftarrow \alpha_1^{i_1} \cdots \alpha_k^{i_k}\, \mathcal{N}$
47 $Y \leftarrow C \oplus \Delta$
48 $X \xleftarrow{\$} \{0,1\}^n$
49 **if** $Y \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow$ true
50 $\pi[X] \leftarrow Y$
51 $M \leftarrow X \oplus \Delta$
52 **return** M

Figure 7: Game 3A, obtained by dropping four statements from Game 3. The probability that bad is set to true may be lessened compared to Game 3, but not by more than by $2q^2/2^n$.

$(1 + 2 + \cdots + 2q - 1)/2^n \leq 2q^2/2^n$. We record this for future reference:

$$\Pr[A^{\text{Game 3}} \text{ sets } bad\,] \;\;\leq\;\; \Pr[A^{\text{Game 3A}} \text{ sets } bad\,] + \frac{2q^2}{2^n} \tag{11}$$

We now make some changes to Game 3A, changes that have no adversarially-visible effect. See Figure 8. (1) We start of by indexing the calls, numbering them from 1 to q. This lets us to drop the *haveSeen* predicate, replacing it by something equivalent. (2) We move the choice of return values up to the initialization step. (3) On a left oracle-query we define the internal value Y from the return value $Z = C$, as as opposed to defining the return value from Y. (4) On a right oracle-query we define the internal value X from the return value $Z = M$, as opposed to defining the return value from X. (5) We delay the computation of *bad* until a finalization step that runs after the adversary has asked its queries. The choices above do not impact the probability that *bad* gets set

Initialization:

Let $\pi[\cdot]$ be everywhere undefined

$Z^1, \ldots, Z^q \xleftarrow{\$} \{0,1\}^n$

$bad \leftarrow$ true

If the r-th query is a left query (T^r, M^r):

return Z^r

If the r-th query is a right query (T^r, C^r):

return Z^r

Finalization:

for $r \leftarrow 1$ **to** q **do**

  if the r-th query was a left query (T^r, M^r) **then**
    Parse T^r into $(b^r, N^r, i_1^r, \ldots, i_k^r)$
    if $N^r = N^p$ for some $p < r$ **then** $\mathcal{N}^r \leftarrow \pi[N^r]$ **else**
      $\mathcal{N}^r \xleftarrow{\$} \{0,1\}^n$
      if $\pi[N^r] \neq$ undefined **then** $bad \leftarrow$ true
      $\pi[N^r] \leftarrow \mathcal{N}^r$
    $\Delta^r \leftarrow \alpha^{i_1^r} \cdots \alpha^{i_k^r} \mathcal{N}^r$
    $X^r \leftarrow M^r \oplus \Delta^r$
    if $X^r \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow$ true
    if $b^r = 0$ **then** $Y^r \leftarrow Z^r$
    if $b^r = 1$ **then** $Y^r \leftarrow Z^r \oplus \Delta^r$
    $\pi[X^r] \leftarrow Y^r$

  if the r-th query was a right query (T^r, C^r) **then**
    Parse T^r into $(b^r, N^r, i_1^r, \ldots, i_k^r)$
    if $N^r = N^p$ for some $p < r$ **then** $\mathcal{N}^r \leftarrow \pi[N^r]$ **else**
      $\mathcal{N}^r \xleftarrow{\$} \{0,1\}^n$
      if $\pi[N^r] \neq$ undefined **then** $bad \leftarrow$ true
      $\pi[N^r] \leftarrow \mathcal{N}^r$
    $\Delta^r \leftarrow \alpha^{i_1^r} \cdots \alpha^{i_k^r} \mathcal{N}^r$
    $Y^r \leftarrow C^r \oplus \Delta^r$
    if $Y^r \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow$ true
    $X^r \leftarrow Z^r \oplus \Delta^r$
    $\pi[X^r] \leftarrow Y^r$

Figure 8: Definition of Game 3B, which sets flag *bad* with the same probability as in Game 3A.

to true and, in particular,

$$\Pr[A^{\text{Game 3A}} \text{ sets } bad\,] \;=\; \Pr[A^{\text{Game 3B}} \text{ sets } bad\,] \tag{12}$$

and our task has been reduced to bounding the probability that *bad* gets set to true at in Game 3B.

At this point we make the observation that, in Game 3B, the association of domain points to range points that is maintained by π is of no significance beyond the "bookkeeping" that π does for recording which values are in the domain of π and which values are in the range of π and what random value is associated to each $\pi[N^r]$. We could just as well have collected up all the domain points which get added to π in a multiset $\mathcal{X}$, and looked for collision, and gathered up all the range points which get added to π in a multiset $\mathcal{Y}$, and looked for collisions. As such, we can eliminate π,

Initialization:
10 $Z^1, \ldots, Z^q \xleftarrow{\$} \{0,1\}^n$

If the r-th query is a left query (T^r, M^r):
20 **return** Z^r

If the r-th query is a right query (T^r, C^r):
30 **return** Z^r

Finalization:
40 **for** $r \in [1\,..\,q]$ **do** parse T^r into $(b^r, N^r, i_1^r, \ldots, i_k^r)$
41 Let $N_1, \ldots, N_p$ be the distinct strings from $N^1, \ldots, N^q$
42 **for** $i \in [1\,..\,p]$ **do** $\mathcal{N}_{N_i} \xleftarrow{\$} \{0,1\}^n$

50 **for** $r \in [1\,..\,q]$ **do**

60 $\quad \Delta^r \leftarrow \alpha_1^{i_1^r} \cdots \alpha_k^{i_k^r} \mathcal{N}_{N^r}$

70 $\quad$ **if** the r-th query was a left query (T^r, M^r) **then**
71 $\quad\quad X^r \leftarrow M^r \oplus \Delta^r$
72 $\quad\quad$ **if** $b^r = 0$ **then** $Y^r \leftarrow Z^r$
73 $\quad\quad$ **if** $b^r = 1$ **then** $Y^r \leftarrow Z^r \oplus \Delta^r$

80 $\quad$ **if** the r-th query was a right query (T^r, C^r) **then**
81 $\quad\quad X^r \leftarrow Z^r \oplus \Delta^r$
82 $\quad\quad Y^r \leftarrow C^r \oplus \Delta^r$

90 $\mathcal{X} \leftarrow [N_1, \ldots, N_p,\ X^1, \ldots, X^q]$
91 $\mathcal{Y} \leftarrow [\mathcal{N}_{N_1}, \ldots, \mathcal{N}_{N_p},\ Y^1, \ldots, Y^q]$

92 $bad \leftarrow$ (there is a repetition in $\mathcal{X}$) **or** (there is a repetition in $\mathcal{Y}$)

Figure 9: Definition of Game 3C, which is adversarially indistinguishable from Game 3B.

accomplish the bookkeeping differently, and check for what would have been collision in the domain or range of π at the very end. All of this is done in Game 3C, show in in Figure 9. Though it looks quite different from Game 3B it sets *bad* under exactly the same circumstances, so

$$\Pr[A^{\text{GameB}} \text{ sets } bad\,] \;=\; \Pr[A^{\text{GameC}} \text{ sets } bad\,] \tag{13}$$

and our task is to evaluate the probability that *bad* gets set in Game 3C.

In Game 3C the adversary asks a sequence of questions and gets back a sequence of random answers. It can only help the adversary if we were to give it all of the answers in advance. The adversary is then at liberty to choose its questions in a way that depends on knowledge of future answers. Since our adversary is deterministic the probability that it sets *bad* to true is over the random values $Z^1, \ldots, Z^q$ returned to the adversary and the random values $\mathcal{N}_1, \ldots, \mathcal{N}_p$ selected as the game runs. We now make the stronger claim that for *any* sequence of responses $Z^1, \ldots, Z^q$ to the adversary's queries, the probability that *bad* gets set to true is small (the probability now being over only the $\mathcal{N}_1, \ldots, \mathcal{N}_p$ values). Once we have fixed $Z^1, \ldots, Z^q$ the adversary's own queries are all fixed, as is whether each query is a left oracle query or a right oracle query. What we aim to show, then, is that for any sequence of valid queries and responses $(b^1, N^1, i_1^1, \ldots, i_k^1, M^1, C^1, Z^1, lr^1), \ldots, (b^q, N^q, i_1^q, \ldots, i_k^q, M^q, C^q, Z^q, lr^q)$ the probability that *bad* will get set to true is small. Here lr^r is an indication if the r-th query was a left oracle query (to encipher, coded as 0) or a right

10 Let $\mathsf{N}_1, \ldots, \mathsf{N}_p$ be the distinct strings from $\mathsf{N}^1, \ldots, \mathsf{N}^q$
11 **for** $i \in [1 \,..\, p]$ **do** $\mathcal{N}_{\mathsf{N}_i} \xleftarrow{\$} \{0,1\}^n$

12 **for** $r \in [1 \,..\, q]$ **do**

20 **if** $\mathsf{lr}^r = 0$ **then**
21 $X^r \leftarrow \mathsf{M}^r \oplus \alpha_1^{\mathsf{i}_1^r} \cdots \alpha_k^{\mathsf{i}_k^r} \mathcal{N}_{\mathsf{N}^r}$
22 $Y^r \leftarrow \mathsf{Z}^r \oplus \mathsf{b}^r \alpha_1^{\mathsf{i}_1^r} \cdots \alpha_k^{\mathsf{i}_k^r} \mathcal{N}_{\mathsf{N}^r}$

30 **if** $\mathsf{lr}^r = 1$ **then**
31 $X^r \leftarrow \mathsf{Z}^r \oplus \alpha_1^{\mathsf{i}_1^r} \cdots \alpha_k^{\mathsf{i}_k^r} \mathcal{N}_{\mathsf{N}^r}$
32 $Y^r \leftarrow \mathsf{C}^r \oplus \alpha_1^{\mathsf{i}_1^r} \cdots \alpha_k^{\mathsf{i}_k^r} \mathcal{N}_{\mathsf{N}^r}$

40 $\mathcal{X} \leftarrow [\mathsf{N}_1, \ldots, \mathsf{N}_p,\ X^1, \ldots, X^q]$
41 $\mathcal{Y} \leftarrow [\mathcal{N}_{\mathsf{N}_1}, \ldots, \mathcal{N}_{\mathsf{N}_p},\ Y^1, \ldots, Y^q]$

50 $bad \leftarrow$ (there is a repetition in $\mathcal{X}$) **or** (there is a repetition in $\mathcal{Y}$)

Figure 10: Definition of Game 3D. All interaction and adaptivity has been eliminated, effectively replaced by universal quantification over the constants (in sans serif font, apart from $p, q, \alpha_1, \ldots, \alpha_k$).

oracle query (to decipher, coded as 1). Thus we fix constants $\mathsf{b}^1, \mathsf{N}^1, \mathsf{i}_1^1, \ldots, \mathsf{i}_k^1, \mathsf{M}^1, \mathsf{C}^1, \mathsf{Z}^1, \mathsf{lr}^1, \ldots, \mathsf{b}^q, \mathsf{N}^q, \mathsf{i}_1^q, \ldots, \mathsf{i}_k^q, \mathsf{M}^q, \mathsf{C}^q, \mathsf{Z}^q, \mathsf{lr}^q)$ and look at the probability that *bad* gets set to true for this vector of constants. The vector of constants must be *valid*, in the sense that the assumptions that we have made about adversarial behaviors are reflected in the constants: no repeated queries; no deciphering when the resulting plaintext is known because of a prior enciphering; and no enciphering when the resulting ciphertext is known because of a prior deciphering. In addition, we insist that the adversary select constants $\mathsf{Z}^1, \ldots, \mathsf{Z}^q$ that are all distinct and that each Z^s differs from M^r and C^r for all $r < s$. This assumption entails a possible decrease in the probability that *bad* gets set to true by at most $1.5\, q^2/2^n$. Call a collection of constants as above *valid*. Now fix a vector of valid constants that maximizes the probability that *bad* will get set to true in Game 3C and regard those constants as fixed. The resulting game, which we call Game 3D, is shown in Figure 10. We have then that

$$\Pr[A^{\text{Game 3C}} \Rightarrow 1] \quad \leq \quad \Pr[\text{Game 3D sets } bad\,] + \frac{1.5q^2}{2^n} \tag{14}$$

and our job is now to bound $\Pr[\text{Game 3D sets } bad\,]$.

We proceed to analyze the probability that *bad* gets set in Game 3D Let us first evaluate the probability of a collision within $\mathcal{X}$,

$$\mathcal{X} = [\mathsf{N}_1, \ldots, \mathsf{N}_p,\ X^1, \ldots, X^q]\,.$$

There can be no collision among the N_r-values because they are, by definition, all distinct. What about collision between an N_s and an X^r? If $\mathsf{lr}^r = 0$ then we are considering

$$\Pr\left[\mathsf{N}_s = \mathsf{M}^r \oplus \alpha_1^{\mathsf{i}_1^r} \cdots \alpha_k^{\mathsf{i}_k^r} \mathcal{N}_{\mathsf{N}^r}\right]$$

which clearly occurs with probability 2^{-n}. If $\mathsf{lr}^r = 1$ then we are considering

$$\Pr\left[\mathsf{N}_s = \mathsf{Z}^r \oplus \alpha_1^{\mathsf{i}_1^r} \cdots \alpha_k^{\mathsf{i}_k^r} \mathcal{N}_{\mathsf{N}^r}\right]$$

which likewise occurs with probability 2^{-n}. What about collision between an X^r and an X^s, where $r < s$? Then depending on lr^r and lr^s we are considering one of:

$$\Pr\left[\mathsf{M}^r \oplus \alpha_1^{\mathsf{i}_1^r} \cdots \alpha_k^{\mathsf{i}_k^r} \mathcal{N}_{\mathsf{N}^r} = \mathsf{M}^s \oplus \alpha_1^{\mathsf{i}_1^s} \cdots \alpha_k^{\mathsf{i}_k^s} \mathcal{N}_{\mathsf{N}^s}\right]$$
$$\Pr\left[\mathsf{M}^r \oplus \alpha_1^{\mathsf{i}_1^r} \cdots \alpha_k^{\mathsf{i}_k^r} \mathcal{N}_{\mathsf{N}^r} = \mathsf{Z}^s \oplus \alpha_1^{\mathsf{i}_1^s} \cdots \alpha_k^{\mathsf{i}_k^s} \mathcal{N}_{\mathsf{N}^s}\right]$$
$$\Pr\left[\mathsf{Z}^r \oplus \alpha_1^{\mathsf{i}_1^r} \cdots \alpha_k^{\mathsf{i}_k^r} \mathcal{N}_{\mathsf{N}^r} = \mathsf{M}^s \oplus \alpha_1^{\mathsf{i}_1^s} \cdots \alpha_k^{\mathsf{i}_k^s} \mathcal{N}_{\mathsf{N}^s}\right]$$
$$\Pr\left[\mathsf{Z}^r \oplus \alpha_1^{\mathsf{i}_1^r} \cdots \alpha_k^{\mathsf{i}_k^r} \mathcal{N}_{\mathsf{N}^r} = \mathsf{Z}^s \oplus \alpha_1^{\mathsf{i}_1^s} \cdots \alpha_k^{\mathsf{i}_k^s} \mathcal{N}_{\mathsf{N}^s}\right]$$

Clearly these are 2^{-n} if $\mathcal{N}^r \neq \mathcal{N}^s$. If $\mathcal{N}^r = \mathcal{N}^s$, however, we have to look at:

$$\Pr\left[\mathsf{M}^r \oplus \mathsf{M}^s = \left(\alpha_1^{\mathsf{i}_1^r} \cdots \alpha_k^{\mathsf{i}_k^r} \oplus \alpha_1^{\mathsf{i}_1^s} \cdots \alpha_k^{\mathsf{i}_k^s}\right) \mathcal{N}_{\mathsf{N}^r}\right]$$
$$\Pr\left[\mathsf{M}^r \oplus \mathsf{Z}^s = \left(\alpha_1^{\mathsf{i}_1^r} \cdots \alpha_k^{\mathsf{i}_k^r} \oplus \alpha_1^{\mathsf{i}_1^s} \cdots \alpha_k^{\mathsf{i}_k^s}\right) \mathcal{N}_{\mathsf{N}^r}\right]$$
$$\Pr\left[\mathsf{Z}^r \oplus \mathsf{M}^s = \left(\alpha_1^{\mathsf{i}_1^r} \cdots \alpha_k^{\mathsf{i}_k^r} \oplus \alpha_1^{\mathsf{i}_1^s} \cdots \alpha_k^{\mathsf{i}_k^s}\right) \mathcal{N}_{\mathsf{N}^r}\right]$$
$$\Pr\left[\mathsf{Z}^r \oplus \mathsf{Z}^s = \left(\alpha_1^{\mathsf{i}_1^r} \cdots \alpha_k^{\mathsf{i}_k^r} \oplus \alpha_1^{\mathsf{i}_1^s} \cdots \alpha_k^{\mathsf{i}_k^s}\right) \mathcal{N}_{\mathsf{N}^r}\right]$$

For all of these, if $(\mathsf{i}_1^r, \ldots, \mathsf{i}_k^r) \neq (\mathsf{i}_1^s, \ldots, \mathsf{i}_k^s)$ then, by the uniqueness of representation for $\alpha_1, \ldots, \alpha_k$ over $\mathbb{I}_1 \times \cdots \times \mathbb{I}_k$, the value multiplying $\mathcal{N}_{\mathsf{N}^r}$ is nonzero and the indicated probability is 2^{-n}. We have left to consider the case of $(\mathsf{N}^r, \mathsf{i}_1^r, \ldots, \mathsf{i}_k^r) = (\mathsf{N}^s, \mathsf{i}_1^s, \ldots, \mathsf{i}_k^r)$, whence we are looking at

$$\Pr[\mathsf{M}^r = \mathsf{M}^s]$$
$$\Pr[\mathsf{M}^r = \mathsf{Z}^s]$$
$$\Pr[\mathsf{Z}^r = \mathsf{M}^s]$$
$$\Pr[\mathsf{Z}^r = \mathsf{Z}^s]$$

Enciphering queries may not be repeated, so the first probability is zero. We have insisted in our choice of valid constants that $M^r \neq Z^s$, so the second probability is zero. An adversary that makes a deciphering query with a given tweak is not allowed to subsequently make an enciphering query of the resulting answer with the same tweak, so the third probability is zero. And deciphering queries may not be repeated, so the forth probability is zero. We may conclude that the probability that there is a collision in $\mathcal{X}$ is at most $2q^2/2^n$.

We next bound the probability of a collision in the multiset

$$\mathcal{Y} = [\mathcal{N}_{\mathsf{N}_1}, \ldots, \mathcal{N}_{\mathsf{N}_p}, Y^1, \ldots, Y^q] .$$

For $r < s$ the probability of a collision between an $\mathcal{N}_{\mathsf{N}_r}$ and an $\mathcal{N}_{\mathsf{N}_s}$ is 2^{-n} since these are random and independent n-bit strings. What about a collision between an $\mathcal{N}_{\mathsf{N}_s}$ and a Y^r? If $\mathsf{lr}^r = 0$ we are considering

$$\Pr\left[\mathcal{N}_{\mathsf{N}_s} = \mathsf{Z}^r \oplus \mathsf{b}^r \alpha_1^{\mathsf{i}_1^r} \cdots \alpha_k^{\mathsf{i}_k^r} \mathcal{N}_{\mathsf{N}^r}\right] .$$

If $\mathsf{N}_s \neq \mathsf{N}^r$ then this probability is 2^{-n}. If $\mathsf{N}_s = \mathsf{N}^r$ and $\mathsf{b}^r = 0$ then the probability is 2^{-n}. If $\mathsf{N}_s = \mathsf{N}^r$ and $\mathsf{b}^r = 1$ then the probability we are considering is

$$\Pr\left[\mathsf{Z}^r = \left(1 \oplus \alpha_1^{\mathsf{i}_1^r} \cdots \alpha_k^{\mathsf{i}_k^r}\right) \mathcal{N}_{\mathsf{N}^r}\right] .$$

By the unique-representation property and the prohibition of the vector of indices $(\mathsf{i}_1^r, \ldots, \mathsf{i}_k^r) = (0, \ldots, 0)$ the parenthesized quantity is not zero and the indicated probability is 2^{-n}. Continuing, a collision between an $\mathcal{N}_{\mathsf{N}_s}$ and a Y^r can also occur when $\mathsf{lr}^r = 1$, in which case we are looking at

$$\Pr\left[\mathcal{N}_{\mathsf{N}_s} = \mathsf{C}^r \oplus \alpha_1^{\mathsf{i}_1^r} \cdots \alpha_k^{\mathsf{i}_k^r}\, \mathcal{N}_{\mathsf{N}^r}\right] .$$

If $\mathsf{N}_s \neq \mathsf{N}^r$ then the above is 2^{-n}. Otherwise we are looking at

$$\Pr\left[\mathsf{C}^r = \left(1 \oplus \alpha_1^{\mathsf{i}_1^r} \cdots \alpha_k^{\mathsf{i}_k^r}\right) \mathcal{N}_{\mathsf{N}^r}\right]$$

and the parenthesized quantity is nonzero for the same reason as before, making the probability 2^{-n}. Finally, we must look at the probability that $Y^r = Y_s$. Then depending on the values of lr^r and lr^s we are considering one of:

$$\begin{aligned}
&\Pr\left[\mathsf{Z}^r \oplus \mathsf{b}^r \alpha_1^{\mathsf{i}_1^r} \cdots \alpha_k^{\mathsf{i}_k^r}\, \mathcal{N}_{\mathsf{N}^r} = \mathsf{Z}^s \oplus \mathsf{b}^s \alpha_1^{\mathsf{i}_1^s} \cdots \alpha_k^{\mathsf{i}_k^s}\, \mathcal{N}_{\mathsf{N}^s}\right] \\
&\Pr\left[\mathsf{Z}^r \oplus \mathsf{b}^r \alpha_1^{\mathsf{i}_1^r} \cdots \alpha_k^{\mathsf{i}_k^r}\, \mathcal{N}_{\mathsf{N}^r} = \mathsf{C}^s \oplus \alpha_1^{\mathsf{i}_1^s} \cdots \alpha_k^{\mathsf{i}_k^s}\, \mathcal{N}_{\mathsf{N}^s}\right] \\
&\Pr\left[\mathsf{C}^r \oplus \alpha_1^{\mathsf{i}_1^r} \cdots \alpha_k^{\mathsf{i}_k^r}\, \mathcal{N}_{\mathsf{N}^r} = \mathsf{Z}^s \oplus \mathsf{b}^s \alpha_1^{\mathsf{i}_1^s} \cdots \alpha_k^{\mathsf{i}_k^s}\, \mathcal{N}_{\mathsf{N}^s}\right] \\
&\Pr\left[\mathsf{C}^r \oplus \alpha_1^{\mathsf{i}_1^r} \cdots \alpha_k^{\mathsf{i}_k^r}\, \mathcal{N}_{\mathsf{N}^r} = \mathsf{C}^s \oplus \alpha_1^{\mathsf{i}_1^s} \cdots \alpha_k^{\mathsf{i}_k^s}\, \mathcal{N}_{\mathsf{N}^s}\right]
\end{aligned}$$

The first of these is 0 if $\mathsf{b}^r = \mathsf{b}^s = 0$ because we have arranged that Z^i-values are all distinct. It is 2^{-n} if one of b^r and b^s is zero and the other is 1. Otherwise, $\mathsf{b}^r = \mathsf{b}^s = 1$. The probability above is clearly 2^{-n} if $\mathcal{N}^r \neq \mathbb{N}^s$, while if is the same value when $\mathcal{N}^r = \mathcal{N}^s$ by the unique-representation property. The remaining three probabilities are all at most 2^{-n} by reasoning exactly analogous to that which has been given. We conclude that the probability that there is a collision in $\mathcal{Y}$ is at most $2q^2/2^n$. Summing up, we have that

$$\Pr[\text{Game 3D sets } \mathit{bad}] \;\leq\; \frac{4q^2}{2^n} \tag{15}$$

and, combining everything, the advantage of our original adversary A is at most $\mathbf{Adv}_E^{\widetilde{\text{prp}}}(t' + 2q) + (2 + 2 + 1.5 + 4)q^2/2^n$, establishing our result.

C Proof of Theorem 11 — Security of OCB1

Consulting Figure 2 may be helpful. In particular, look at the figure on the top and understand that each π_i^N and $\bar{\pi}_i^N$ is a random permutation on n bits. All of these permutations are independent. In effect, the key is the infinite collection of random permutations π_i^N, $\bar{\pi}_i^N$ for $i \in \mathbb{N}$ and $N \in \{0,1\}^n$.

The privacy statement is immediate. During the adversary's attack it asks a sequence of queries $(N^1, M^1), \ldots, (N^q, M^q)$ where the N^i-values are distinct. Since the N^i-values are distinct each π_i^N and $\bar{\pi}_i^N$ that gets used gets used exactly once. We are thus applying a number of independent random permutations each to a *single* point. The image of a single point under a random permutation is uniform, so the output is perfectly uniform. That is all that is needed to ensure that an adversary has no advantage to distinguish the output from random bits.

The authenticity statement is more involved, but still straightforward. Before we launch into it, consider the following simple game. Suppose that you know that an n-bit string X is *not* some particular value X_0. All of the $2^n - 1$ other values are equally likely. Then your chance of correctly

predicting the τ-bit prefix of X is at most $2^{n-\tau}/(2^n-1)$. That's because the best strategy is to guess any τ-bit string other than the τ-bit prefix of X_0. The probability of being right under this strategy is $2^{n-\tau}/(2^n-1)$. We will use this fact in the sequel.

Now suppose that the adversary asks a sequence of queries $(N^1, M^1), \ldots, (N^q, M^q)$ and then makes its forgery attempt $(N, \mathcal{C})$. Let $M^i = M^i_1 \cdots M^i_{m_i}$ be the queries and let $C^i = C^1_1 \cdots C^i_{m_i}$ be the responses. Let the tags produced as the adversary asks its queries be $\mathit{Pad}^1, \ldots, \mathit{Pad}^q$ and let the checksums produced be $\Sigma^1, \ldots, \Sigma^q$. Let $\mathcal{C} = C \parallel T$ where T is the first τ bits of Tag and $C = C_1 \ldots C_c$. Let the pad and checksum for the forgery attempt be Pad and Σ. We consider a number of cases.

(1) Suppose $N \notin \{N^1, \ldots, N^q\}$—the adversary tries to forge using a new nonce. Then the adversary needs to find the correct value of T but has seen no image of the random permutation $\bar{\pi}^N_c$. The chance that the adversary can guess the correct value for a random τ-bit string, given no information about it, is $2^{-\tau}$.

(2) Suppose $N = N^i$ but $c \neq m_i$. As above, the adversary needs to find the correct value of T but has seen no image of the random permutation $\bar{\pi}^N_c$. Thus the chance that the adversary can guess the correct value is $2^{-\tau}$.

(3) Suppose $N = N^i$ and $c = m_i$ but $|C| \neq |M^i|$. First, we may ignore the queries other than the i^{th} since their answers are unrelated to the adversary's task of producing a valid ciphertext $(N, \mathcal{C})$ with $N = N^i$. This time the adversary has seen one point of relevance to determining Pad, namely, the adversary may have seen some or all of Pad^i. No other values returned to the adversary are correlated to π^N_c. Among the possible values of Pad, all but Pad^i are equally likely, so there are $2^n - 1$ equally plausible values for Pad. Thus there are $2^n - 1$ equally plausible values for Σ even if we make public to the adversary all permutations other than π^N_c. Even then there are $2^n - 1$ equally likely inputs to $\bar{\pi}^N_c$ so there are at least $2^n - 1$ equally plausible outputs as Tag. By the analysis at the beginning of the authenticity analysis, the adversary's chance of correctly guessing T is thus at most $2^{n-\tau}/(2^n-1)$.

(4) Suppose $N = N^i$ and $|C| = |M^i|$ and $C_j \neq C^i_j$ for some $j \in [1..c-1]$. We may again ignore the queries other than the i^{th} since their answers are unrelated to the adversary's task of producing a valid ciphertext $(N, \mathcal{C})$ with $N = N^i$. The adversary has seen the preimage under π^N_j of the point C^i_j, namely M^i_j, but the preimage under π^N_j of every point other than C^i_j is equally possible. In particular, the preimage of C_j under π^N_j is equally likely to be any of $2^n - 1$ different strings string. We imagine providing to the adversary all permutations other than the permutation π^N_j. Even then, the value of Σ can be any of $2^n - 1$ strings, each with equal probability. By the analysis at the beginning of the authenticity analysis, the adversary's chance of correctly guessing T is thus at most $2^{n-\tau}/(2^n-1)$.

(5) Finally, suppose $N = N^i$ and $|C| = |M^i|$ and $C_j = C^i_j$ for all $j \in [1..c-1]$ but $C_c \neq C^i_c$. In this case the value Σ is known and differs from Σ^i by the non-zero value $C_c \oplus C^i_c$. The value Tag can be any of $2^n - 1$ values, each with equal likelihood. By the analysis at the beginning of the authenticity analysis, the adversary's chance of correctly guessing T is at most $2^{n-\tau}/(2^n-1)$.

This completes the proof.