A preliminary version of this papers appears in *Advances in Cryptology – ASIACRYPT '00*, Lecture Notes in Computer Science Vol. 1976, T. Okamoto ed., Springer-Verlag, 2000. This is the full version.

Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography

MIHIR BELLARE[*] PHILLIP ROGAWAY[†]

December 2000

Abstract

We investigate the following approach to symmetric encryption: first *encode* the message via some keyless transform, and then *encipher* the encoded message, meaning apply a permutation F_K based on a shared key K. We provide conditions on the encoding functions and the cipher which ensure that the resulting encryption scheme meets strong privacy (eg. semantic security) and/or authenticity goals. The encoding can either be implemented in a simple way (eg. prepend a counter and append a checksum) or viewed as modeling existing redundancy or entropy already present in the messages, whereby encode-then-encipher encryption provides a way to exploit structured message spaces to achieve compact ciphertexts.

Keywords: Ciphers, Encoding, Modes of Operation, Provable Security, Symmetric Encryption.

[*] Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093, USA. E-Mail: `mihir@cs.ucsd.edu`. URL: `www-cse.ucsd.edu/users/mihir`

[†] Dept. of Computer Science, Engineering II Building, One Shields Avenue, University of California at Davis, Davis, CA 95616, USA, and Dept. of Computer Science, Faculty of Science, Chiang Mai University, Thailand. E-Mail: `rogaway@cs.ucdavis.edu`. URL: `www.cs.ucdavis.edu/~rogaway`

Contents

1 Introduction

Enciphering vs. encrypting. Many popular books on cryptography describe "encryption" as applying a key-indexed permutation F_K to the plaintext M, thereby obtaining the ciphertext $C = F_K(M)$. Yet, if the goal of encryption is privacy (as it is usually assumed to be), then our community has long since recognized that, being deterministic, this realization of encryption cannot possibly achieve the strong security guarantees that one would hope for, namely, semantic security under chosen-plaintext attack and beyond [8, 6, 12]. (For example, if the same message is encrypted twice an adversary will be able to detect this.)

From this point forward, a family of permutations $F = \{F_K\}$ will be called a *cipher.* Applying one of these functions, F_K, is *enciphering* (not encrypting). Applying F_K^{-1} is *deciphering* (not decrypting). In this paper, "good" for an enciphering method means approximating (in the usual ways [10]) a family of random permutations. On the other hand, "good" for an encryption scheme means achieving privacy properties at least as strong as semantic security. As indicated above, good enciphering never, by itself, makes for good encryption.

Despite the last statement, there seems to be a widespread belief that enciphering a message is, somehow, almost as good as encrypting it. When messages are somehow "structured," or the message space has "enough entropy," maybe enciphering does the job. Is there some scientific basis for such a belief?

In this paper we investigate the circumstances under which good enciphering really *does* make for good encryption. This leads us to introduce *encoding schemes* as a way to conceptualize what is happening when you encipher structured messages. Let us describe what are encoding schemes, and how they relate to enciphering.

Encode-then-encipher encryption. Start with a good cipher that operates on messages of any length at all. (In other words, F_K, for a random K, "looks like" a random length-preserving permutation.) Now to encrypt M, first "encode" it into some string M^*. The encoding might be extremely simple—like prepending a counter, or appending some 0-bits, or maybe doing both. The encoding might even be the identity function. All that is demanded of an encoding method is that it does not "lose" information: you can "decode" M^* to recover M, and you can recognize when a string is and is not the encoding of any message. Now to encrypt message M under key K, encipher the encoded message M^* using F_K, yielding ciphertext $C = F_K(M^*)$. To decrypt a ciphertext C decipher it to find $M^* = F_K^{-1}(C)$, and then decode M^* to get either a message M or an indication that M^* is not the encoding of any message. We call this style of encryption "encode-then-encipher encryption." This is not a popular way to encrypt, though it is certainly a very natural paradigm.

Our results. In this paper we investigate how properties of the encoding scheme and the enciphering scheme can give rise to security properties of the resulting encryption scheme.

Suppose first that the encoding scheme adds in a *nonce*—usually a counter or a random value. The nonce can be added into the message in any way at all. All one needs is that the "collision probability"—the chance that two encoded messages come out the same"—be small. We prove in Theorem 4.1 that enciphering such encodings provides *semantic security.*

Next we look at encoding schemes which result in encoded messages which have enough *redundancy.* This means that "most" strings M^* will be considered "bad." We prove in Theorem 4.2 that the resulting encryption scheme will now achieve *message authenticity.* It is as though the sender had sent a MAC along with his transmission. Interestingly, this theorem requires that the cipher be a *strong* pseudorandom permutation [10]. We show in Theorem 4.3 that an ordinary pseudorandom permutation won't do.

The actual results are quantitative. They show how much privacy and authenticity is guaranteed as a function of (easily-calculated) numbers associated to the encoding scheme, and as a function of the (quantified) security of the underlying cipher.

Justifying some old intuition. At some level it would seem to be folklore that enciphering strings which employ nonces or redundancy makes for good encryption. In the security literature one sees many statements to the effect that *we assume that messages to be encrypted employ adequate redundancy*, or *we avoid replay attacks by including a nonce in the messages we encrypt.* Our results help formalize what such authors may have had in mind, since the statements above become meaningful and true when "encryption" means "enciphering" and when the roles of nonces and redundancy are formally defined.

Is the encoding step "real"? In some applications of encode-then-encipher encryption we imagine that the encoding step will be an ostensible part of encrypting: the piece of software which encrypts M will encode it first, and then encipher the encoded message. For example, the encryption engine might take in a message M, prepend a counter, append a checksum, and encipher the resulting string. But encode-then-encipher encryption is actually more interesting when the encoding and decoding operations do *not* occur within the customary boundary of the encryption engine. For example, the encryption software may be presented with an already-formatted IP packet M^*. Its payload is the message M one should get on decoding M^*, but the encryption software itself knows nothing about where is the payload or how to extract it. Still, the encoding and decoding processes really did occur, albeit within a different piece of code. Finally, the encoding step may exist purely as a conceptualization. For example, if messages are supposed to be English-language sentences then the encoding step can be regarded as the the identity function on the space of proper English-language sentences, while the decoding function takes a string M^* and returns $M = M^*$ if it is English, or else an indication that this is not an English sentence. Probably this decoding operation can only performed by a human! Nonetheless, even in this case the language of encodings makes sense.

In general, the encoding of messages should be seen as a *model* for how the messages that we are enciphering might arise. This model is a more useful and general approach than trying to equip an unknown message space with a distribution. For example, a distribution on messages can not handle ideas like inserting a counter into the message, and it is quite artificial to try to equip English-language utterances with some distribution. The encoding/decoding model lets us discuss, in a natural and simple way, all the relevant properties about how messages might look.

Why encode-then-encipher? Encode-then-encipher encryption can be used to provide short ciphertexts with a high degree of independence on message-formatting conventions. As such, it can be used to provide a convenient migration path for legacy protocols. Let us explain.

In various application, particularly in networking, a "packet format" will have been defined, where this packet format includes redundancy and/or nonces, but has no fields for cryptographic purposes (eg., fields for an IV or MAC). Now suppose a need arises to add in privacy or authenticity features. At the same time, there will often be a real-world constraint not to grow or re-define the packet format.

Using encode-then-encipher you probably do not have to. If packets are known to repeat rarely or not at all (eg., packets always contain a sequence number) then semantic security is automatically guaranteed just by applying a good cipher. And if packet formats already include redundancy (which they typically do if for no other reason than to simplify parsing) then there may be no need to add in a separate MAC; once again, good enciphering (this time, with a *strong* pseudorandom permutation) is enough. And because it is irrelevant how and where the nonce and redundancy appeared in the packet, privacy and authenticity will be retained, with no protocols changes at all, if packet formats should subsequently change in some details.

The result is that encode-then-encipher encryption would leave packet sizes alone (our ciphers are understood to be length-preserving), and they would leave packets looking identical (after deciphering) to the way they looked before. This allows for modular software changes with minimal code disruption. The code which enciphers as a way to encrypt doesn't know (or care) where is the sequence number, say, where other fields are, or what values these fields can take. Such indifference makes for robust and simple software, and thus an easier migration path for adding in security features.

Constructing variable-input-length ciphers. To encrypt messages using the encode-then-encipher approach you need to encipher strings which may be long or short, and whose lengths may vary from one enciphering to the next. The cipher should look like a random length-preserving permutation $\pi : \mathcal{M}^* \to \mathcal{M}^*$. This may sound just like a block cipher, but it is actually quite different, because the domain includes strings of different lengths. One construction is given in [5], and others are possible, building on work like [10] and [11].

A notion of authenticity for encryption schemes. We note a final contribution of this paper, which is the notion of authenticity defined in Section 2. The usual way that message authenticity has been defined (eg., [2]) assumes that each message M is accompanied by a tag (the message authentication code) τ. The adversary wants to produce a hitherto unseen message M' and a valid tag τ' for it. But this setting does not apply to us, where the messages being authenticated are never made visible. In the new setting the adversary's goal is to get the receiver to accept as authentic a string C—with a possibly unknown "meaning" M—where the adversary has not already witnessed C. This necessitates a new notion (or measure) of security for a symmetric encryption scheme.

While several definitions of privacy for symmetric encryption schemes are given in [1], here we are suggesting a notion of authenticity for an encryption scheme. Namely, consider a symmetric encryption scheme in which the decryption algorithm is allowed to reject ciphertexts to indicate that they are unauthentic. We take the setting of [1] in which the adversary gets to see (via an oracle) ciphertexts of messages of her choice encrypted under a key K. We then say that the adversary wins if she can produce a valid ciphertext (meaning one which the decryption function under K does not reject) which was never an output of the encryption oracle.

Early (submitted) versions of this paper date to December 1998. Since then, definitions of authenticity for symmetric encryption schemes have appeared elsewhere [9]. We refer the reader to [4] for a comprehensive treatment of different notions of authenticity for symmetric encryption schemes and their relations to the notions of privacy.

2 Definitions

We provide definitions for PRFs, PRPs and SPRPs over arbitrary message spaces, and definitions of privacy and authenticity for symmetric encryption schemes.

History and comparisons. The basic definition of a PRF (pseudorandom function), as given by [7], sets the domain, range and keyspace to be the set of strings of length equal to the security parameter, and then defines security asymptotically. We adopt concrete versions of these definitions, as per [2], in order to model block-cipher based construction, and also to allow for a domain (which we call the message space) containing strings of different lengths. Our notion of a PRP (pseudorandom permutation) follows [2, 3] and differs from that of [10] in that we measure distinguishability versus a random permutation rather than a random function, which is important when concrete security is considered. The notion of an SPRP (strong pseudorandom permutation) is that of [10] concretized in the style of [2] and extended with regard to domains. The definition of privacy for symmetric encryption schemes is from [1].

NOTATION AND CONVENTIONS. A *message space* $\mathcal{M}$ is a subset of $\{0,1\}^*$ for which $x \in \mathcal{M}$ implies that $x' \in \mathcal{M}$ for all x' of the same length of x, and for which there exists an efficient (say linear time) algorithm to decide membership. A *ciphertext space* $\mathcal{C}$ is a subset of $\{0,1\}^*$. A *key space* $\mathcal{K}$ is a set together with a probability measure on that set. Writing $K \leftarrow \mathcal{K}$ means to choose K at random according to this probability measure. The notation $|X|$ denotes the length of X if X is a string and the number of elements in X if X is a set.

CIPHERS. Let $\mathcal{K}$, $\mathcal{M}$ and $\mathcal{C}$ be a key space, message space, and ciphertext space. A *family of functions* is a map $F\colon\ \mathcal{K} \times \mathcal{M} \to \mathcal{C}$. If $K \in \mathcal{K}$ then we let $F_K(\cdot) = F_K(\cdot)$ and call this an *instance* of F. We let $f \leftarrow F$ denote the operation of picking a function from F at random. (This is shorthand for $K \leftarrow \mathcal{K}$; $f \leftarrow F_K$.) We assume that $|F_K(M)| = \ell(|M|)$ depends only on $|M|$ and call ℓ the *length function* of the family. A *cipher* is a family of functions $F\colon\ \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ in which each $F_K : \mathcal{M} \to \mathcal{C}$ is one-to-one and onto. In this case, F_K^{-1} denotes the inverse of $F_K(\cdot)$. A cipher is *length-preserving* if $F_K(M) = |M|$ for all $K \in \mathcal{K}$ and $M \in \mathcal{M}$. For simplicity, all ciphers in this paper are assumed to be length-preserving. A *block-cipher* is a cipher with domain and range $\{0,1\}^n$. The number n is called the *block length*.

We let $\mathbf{Rand}(\mathcal{M},\ell)$ denote the family of all functions $f\colon\ \mathcal{M} \to \{0,1\}^*$ that satisfy $|f(M)| = \ell(|M|)$ for all $M \in \mathcal{M}$. A random function f from $\mathbf{Rand}(\mathcal{M},\ell)$ is determined as follows: for each $M \in \mathcal{M}$, $f(M)$ is a random string of length $\ell(|M|)$. Also let $\mathbf{Perm}(\mathcal{M})$ denote the cipher consisting of all length-preserving, one-to-one and onto functions on $\mathcal{M}$. A random function π from $\mathbf{Perm}(\mathcal{M})$ is determined as follows: for each number i such that $\mathcal{M}$ contains strings of length i, let π_i be a random permutation on $\{0,1\}^i$. Then define $\pi(M) = \pi_i(M)$, where $i = |M|$.

PRFS, PRPS AND SPRPS. A *distinguisher* is a (possibly probabilistic) algorithm A which has access to an oracle. If $F\colon\ \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ is a function family with length function ℓ we let

$$\mathbf{Adv}_F^{\mathrm{prf}}(A) \ = \ \Pr[K \leftarrow \mathcal{K} : A^{F_K(\cdot)} = 1] - \Pr[f \leftarrow \mathbf{Rand}(\mathcal{M},\ell) : A^{f(\cdot)} = 1]$$

denote the advantage of A in distinguishing F from a random function. We let

$$\mathbf{Adv}_F^{\mathrm{prp}}(A) \ = \ \Pr[K \leftarrow \mathcal{K} : A^{F_K(\cdot)} = 1] - \Pr[\pi \leftarrow \mathbf{Perm}(\mathcal{M}) : A^{\pi(\cdot)} = 1]$$

denote the advantage of A in distinguishing F from a random permutation. Define

$$\begin{aligned} \mathbf{Adv}_F^{\mathrm{prf}}(t,q,\mu) &= \max_A\{\mathbf{Adv}_F^{\mathrm{prf}}(A)\} \\ \mathbf{Adv}_F^{\mathrm{prp}}(t,q,\mu) &= \max_A\{\mathbf{Adv}_F^{\mathrm{prp}}(A)\} \end{aligned}$$

where the maximum is taken over all adversaries having time-complexity at most t and asking at most q oracle queries, these queries totaling at most μ bits. (The time-complexity, here and hereafter, refers to the execution time of the experiment underlying the definition of the advantage, plus the size of the description of the adversary.)

To define SPRPs we give the distinguisher not only an oracle for the function, but also one for its inverse. Let $F\colon\ \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ be a PRP with length function ℓ. Then we let

$$\begin{aligned} &\mathbf{Adv}_F^{\mathrm{sprp}}(A) \ = \\ &\qquad \Pr[K \leftarrow \mathcal{K} : A^{F_K(\cdot),F_K^{-1}(\cdot)} = 1] - \Pr[\pi \leftarrow \mathbf{Perm}(\mathcal{M}) : A^{\pi(\cdot),\pi^{-1}(\cdot)} = 1] \end{aligned}$$

denote the advantage of A in distinguishing F from a random permutation. Define

$$\mathbf{Adv}_F^{\mathrm{sprp}}(t,q,\mu) \ = \ \max_A\{\mathbf{Adv}_F^{\mathrm{sprp}}(A)\}$$

where the maximum is taken over all adversaries having time-complexity at most t and asking at most q oracle queries, these queries totaling at most μ bits.

Throughout, if the distinguisher inquires as to the value of oracle f at a point $M \notin \mathcal{M}$ then the oracle responds with the distinguished point $\perp$. Since we assume that there is a (simple) algorithm to decide membership in $\mathcal{M}$ there is in fact no point for the adversary to make such inquiries.

Encapsulation schemes. Fix a key space $\mathcal{K}$, a message space $\mathcal{M}$, and a ciphertext space $\mathcal{C}$. An *encapsulation scheme* $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is a triple of algorithms. The probabilistic key-generation algorithm $\mathcal{K}$ produces a key $K \in \mathcal{K}$; we write $K \leftarrow \mathcal{K}$. The encryption algorithm $\mathcal{E}$ can be either probabilistic or stateful. It takes a key $K \in \mathcal{K}$ and a message $M \in \mathcal{M}$ and returns ciphertext $C = \mathcal{E}_K(M, r) \in \mathcal{C} \cup \{\perp\}$. If probabilistic, $r \in \{0,1\}^*$ is its coins tosses, which are taken anew upon each invocation. If stateful, r is the internal state, which the encryption algorithm updates upon each invocation and which is securely maintained across invocations. (The state is typically a counter, which is incremented by some message-dependent amount.) The value $\perp$ is returned if $M \notin \mathcal{M}$ or (if this is a stateful encryption scheme) the state r indicates that the message M can not be sent (when, for example, too many messages have already been sent). Algorithm $\mathcal{D}$ takes $K \in \mathcal{K}$ and $C \in \{0,1\}^*$ and computes $M = \mathcal{D}_K(C)$ where M is either a string in $\mathcal{M}$ or the distinguished symbol $\perp$. A return value of $\perp$ is used to indicate that C is regarded as unauthentic. We call C *valid* if $\mathcal{D}_K(C) \in \mathcal{M}$ and we call C *invalid* if $\mathcal{D}_K(C) = \perp$. We also permit applying $\mathcal{E}_K$ to $(\perp, r)$, which results in a return value of $\perp$. Likewise, applying $\mathcal{D}_K$ to $\perp$ is permitted and this gives a return value of $\perp$. We require that if $C = \mathcal{E}_K(M, r)$ and $C \neq \perp$ then $\mathcal{D}_K(C) = M$.

When we think of the goal of $\mathcal{SE}$ as privacy, or a combination of privacy and message authenticity, we typically call it an *encryption scheme*. When we think of the goal of $\mathcal{SE}$ as authenticating messages then we call it an *authentication scheme*. But we emphasize that there is no syntactic distinction between an encryption scheme and an authentication scheme under this formalization: they are both encapsulation schemes.

Privacy. Several formulations for the privacy of a symmetric encryption scheme under chosen-plaintext attack were provided in [1] and compared in terms of concrete security. We will use one of these notions, namely "real-or-random" security. The idea is that an adversary cannot distinguish the encryption of text from the encryption of an equal-length string of garbage. For the formalization, let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme and let A be an adversary with an encryption oracle. If the encryption scheme is probabilistic then fresh random choices are made for each query. If the encryption scheme is stateful then the state is properly initialized and then adjusted with each query. Define

$$\mathbf{Adv}^{\mathrm{priv}}_{\mathcal{SE}}(A) \;=\; \Pr\left[K \leftarrow \mathcal{K} : A^{\mathcal{E}_K(\cdot)} = 1\right] - \Pr\left[K \leftarrow \mathcal{K} : A^{\mathcal{E}_K(\$^{|\cdot|})} = 1\right] .$$

In the first game, the oracle, given a message, returns an encryption of it under key K; in the second game the oracle, given a message, ignores it except to record its length n, and then returns an encryption of a random message of length n. The advantage of A is a measure of the adversary's ability to tell these two worlds apart. We let

$$\mathbf{Adv}^{\mathrm{priv}}_{\mathcal{SE}}(t, q, \mu) \;=\; \max_{A}\{\mathsf{Adv}^{\mathrm{priv}}_{\Pi}(A)\}$$

where the maximum is over all adversaries which have time-complexity at most t and ask at most q oracle queries, where these queries total at most μ bits.

Authenticity. Consider parties sharing a key K and sending messages using an encapsulation scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. We are interested in authenticity: the receiver wants to be confident that a received ciphertext (and underlying message) really did originate with the sender. To formalize this an adversary will be given a way to generate authenticated messages of her choice:

$M_1 \mapsto C_1, M_2 \mapsto C_2, \ldots, M_q \mapsto C_q$. She will "win" if she computes a new string C (that is, $C \notin \{C_1, \ldots, C_q\}$) which would be deemed authentic by the receiver.

Authenticity in the context of an encapsulation scheme is a more general concept than that of a message authentication code (MAC). A MAC makes explicit a particular mechanism, namely the attachment of a tag to the transmission. (The tag, computed using the key, is created by the sender and checked by the receiver.) An encapsulation scheme may use a MAC, or may not, and consideration of authenticity for such a scheme cannot make assumptions about the presence of any type of mechanism. But there is a deeper difference between a MAC and a general authentication scheme. In formalizing the security of a MAC the adversary makes a number of queries to a MAC-generation oracle, with each query mapping the message M_i to its tag t_i. After that the adversary has to come up with a new message M and a tag t such that the receiver will deem (M, t) authentic. In particular, the adversary must "know" the message M that is being forged, insofar as the adversary outputs it along with t. In contrast, an adversary attacking an authentication scheme in the general sense we are defining wins *even if she does not know what is the message M which is being forged.* All that is required is that there *is* such a message underlying C—that is, the receiver will recover *something* in the message space $\mathcal{M}$ (and not an indication that C is bogus).

Formally, let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an authentication scheme and let A be an adversary who is given oracle access to $\mathcal{E}$. After interacting with that oracle the adversary outputs a string C. We say C is *new* if C was not the response to any earlier oracle query asked by A. Adversary A is said to be *successful* if C is new and valid, and we measure the probability of this:

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathcal{SE}}(A) \;=\; \Pr[\, K \leftarrow \mathcal{K}\, ; C \leftarrow A^{\mathcal{E}_K(\cdot)} \;:\; C \text{ is new and } \mathcal{D}_K(C) \neq \bot \,]\;.$$

The quality of $\mathcal{SE}$ in authenticating messages is measured by the function

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathcal{SE}}(t, q, \mu) \;=\; \max_{A}\{\mathbf{Adv}^{\mathrm{auth}}_{\Pi}(A)\}$$

where the maximum is over all adversaries who have time-complexity at most t and make at most $q-1$ oracle calls, these totaling a most $\mu - |C|$ bits, where C is the length of A's output. For simplicity, we assume that an adversary A attacking the authenticity of $\mathcal{SE}$ will only output a string which is new.

The above notion is called "integrity of ciphertexts" in [4] who provide a comprehensive picture of how it relates to other notions of privacy and authenticity for encapsulation schemes. In particular they show that integrity of ciphertexts plus privacy against chosen-plaintext attack imply privacy under chosen-ciphertext attack.

3 Encoding Schemes

Syntax. Fix message spaces $\mathcal{M}, \mathcal{M}^*$. An *encoding scheme* ("on $\mathcal{M}$", or "from $\mathcal{M}$ to $\mathcal{M}^*$") is a pair of algorithms $\mathsf{Encode} = (\mathit{Encode}, \mathit{Decode})$ as we now describe.

Algorithm *Encode* can be either probabilistic or stateful, while *Decode* is neither. First assume that *Encode* is probabilistic (not stateful). Then each time *Encode* is called on an input $M \in \mathcal{M}$ the algorithm flips some coins, r, and returns a string $M^* = \mathit{Encode}(M, r) \in \mathcal{M}^*$. We assume that for any string $M \in \mathcal{M}$ and any coins r, we have that $|\mathit{Encode}(M, r)| = \ell(|M|)$ for some function ℓ, the "length function" of the encoding scheme.

Algorithm *Decode* takes as input $M^* \in \{0,1\}^*$. It returns either a binary string $M \in \mathcal{M}$ or the distinguished symbol $\bot$. If $\mathit{Decode}(M^*)$ is a binary string we say that M^* is *valid*, while we say that M^* is *invalid* if $\mathit{Decode}(M^*) = \bot$. We demand that for any $M \in \mathcal{M}$ and any r, we have that $\mathit{Decode}(\mathit{Encode}(M, r)) = M$.

We allow that *Encode* and *Decode* be presented with any string at all, even ones outside of $\mathcal{M}$

and $\mathcal{M}^*$. If you try to encode a string $M \notin \mathcal{M}$ then the result is the distinguished value $\bot$. If you try to decode a string $M^* \notin \mathcal{M}^*$ then the result is the distinguished value $\bot$. We further establish the convention that you can encode or decode $\bot$, which once again returns $\bot$.

For simplicity in theorem statements we assume that *Encode* and *Decode* are efficiently computable, say in linear time.

Rare-collision encodings. Let $\mathsf{Encode} = (\mathit{Encode}, \mathit{Decode})$ be an encoding scheme and let $\ell(n)$ be its length function. Let $\epsilon\colon\ \mathsf{N} \to \mathbf{R}$ be a function. We say that Encode is **ϵ-colliding** if for and any number q and any (even computationally unbounded) adversary A who asks q queries, the probability that some two of these queries receive the same valid response is at most $\epsilon(q)$.

$$\begin{gathered}\Pr[(M_1^*,\ldots,M_q^*) \leftarrow \mathsf{Responses}\ A^{\mathit{Encode}(\cdot)} :\ \exists\ i < j \text{ s.t. } M_i^* \neq \bot, \\ M_j^* \neq \bot, \text{ and } M_i^* = M_j^*] \leq \epsilon(q)\ .\end{gathered}$$

We shall say that $\langle M_1^*, \ldots M_q^* \rangle$ "collide" if some pair of these strings are the same and are different from $\bot$. The reader may prefer to think of $M_1 = M_2 = \cdots = M_q$ since typically this would be the adversary's best strategy when trying to produce a collision (as $M \neq M'$ implies that their encodings, if valid, have to be different).

Example 3.1 Encoding scheme Prepend-128-Random-Bits works as follows. The message space is $\mathcal{M} = \{0,1\}^*$. Function *Encode* takes an input M and outputs $r \parallel M$, where r is a sequence of 128 random bits. Function *Decode* takes an input M^* and behaves as follows. If M^* is at least 128 bits, then *Decode* outputs all but the first 128 bits of M^*. If M^* is less than 128 bits then $\mathit{Decode}(M^*)$ outputs $\bot$. Then Prepend-128-Random-Bits is $C(q, 2^{128})$-colliding, where $C(q,m)$ denotes the probability of at least one collision in the experiment of throwing q balls, independently and at random, into m bins.■

Collision-free encodings. For algorithm *Encode* to be stateful means that it maintains state across invocations. The initial value of that state is some fixed constant, r_0. Typically there will be a limit, N, on the number of times that *Encode* may be used. After that number of invocations *Encode* will return $\bot$ even when the inquiry is in $\mathcal{M}$. We require that for all messages M and all internal states r, if $\mathit{Encode}(M,r)$ returns a binary string M^* then $\mathit{Decode}(M^*) = M$. We emphasize that decoding is stateless.

Stateful encoding schemes are of interest because with them we can make an encoding scheme **collision free**, meaning 0-colliding, in the language above. Note that getting two $\bot$ values does *not* count as a collision. Here is an example.

Example 3.2 Encoding scheme Prepend-64-Bit-Counter works as follows. The message space is $\mathcal{M} = \{0,1\}^*$. A counter ctr is initialized to 0. The i-th message is encoded as follows. If $i \geq 2^{64}$ then the encoding is $\bot$. Otherwise the encoding is $M^* = \langle i \rangle \parallel M$, where $\langle i \rangle$ the number i written as a 64-bit binary string. Function *Decode* takes an input M^* and behaves as follows. If $|M^*| < 64$ then *Decode* returns $\bot$. Otherwise it returns M^* after having expunged the first 64-bits. Clearly Prepend-64-Bit-Counter is collision free: the counter guarantees that no two encodings can collide.■

Sparse encodings. Let $\mathsf{Encode} = (\mathit{Encode}, \mathit{Decode})$ be an encoding scheme and let δ be a real number. We say that encoding scheme Encode is **δ-dense** if for all $n \in \mathsf{N}$,

$$\Pr[\, M^* \leftarrow \{0,1\}^n : \mathit{Decode}(M^*) \in \{0,1\}^* \,] \leq \delta\ .$$

That is, for every message length, at most a δ-fraction of all strings of that length are valid (they decode to strings in $\mathcal{M}$). The rest are invalid encodings (they decode to $\bot$).

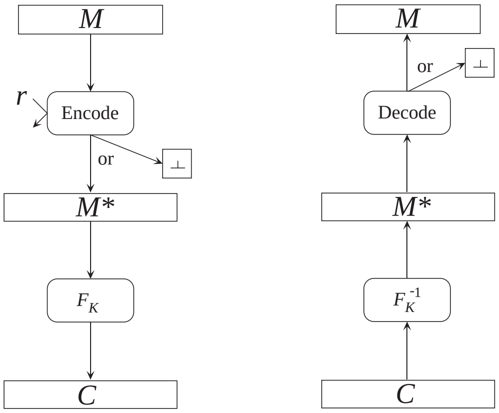

Figure 1: *Scheme* $F \circ \mathsf{Encode}$*: encrypting (left-hand side) and decrypting (right-hand side) using the encode-then-encipher paradigm. The plaintext is* M*, the ciphertext is* C*, the cipher is* $F = \{F_K\}$*, and the encoding scheme is* $\mathsf{Encode} = (\mathit{Encode}, \mathit{Decode})$.

Example 3.3 The encoding scheme $\mathsf{Prepend\text{-}32\text{-}Zeros}$ works as follows. Let $\mathcal{M} = \{0,1\}^*$. Define $\mathit{Encode}(M) = 0^{32} \parallel M$. Define $\mathit{Decode}(M^*)$ to be M^* after stripping away its first 32 bits, assuming that M^* has at least 32 bits, and set $\mathit{Decode}(M^*) = \perp$ otherwise. Then $\mathsf{Prepend\text{-}32\text{-}Zeros}$ is 2^{-32}-dense: a string is valid (it starts with 32 zeros) with probability at most 2^{-32}. Indeed the probability that a random string M^* is valid is exactly 2^{-32} if the length of M^* is at least 32 bits, while the probability is 0 if the length of M^* is less than 32 bits.▮

Example 3.4 Let the message space $\mathcal{M}$ be odd-parity-adjusted ASCII strings of length at least 50 bytes. This means that a message $M \in \mathcal{M}$ is a sequence of bytes $M = b_1 \parallel \cdots \parallel b_n$, for $n \geq 50$, where each b_i is a byte having its low 7 bits arbitrary and its high bit whatever is necessary so that the number of 1-bits in b_i will be odd. Encoding scheme $\mathsf{Odd\text{-}Parity}$ is defined as follows. Function *Encode* is the identity function. Function *Decode* checks that the bit length of its input is divisible by 8, that the input is at least 50 bytes, and that each byte has odd parity. If these conditions are satisfied then *Decode* returns its input. Otherwise it returns $\perp$. Then $\mathsf{Odd\text{-}Parity}$ is 2^{-50}-dense: a random string is valid with probability at most 2^{-50}. Indeed the probability that a random n-byte string is valid is 2^{-n} if $n \geq 50$, and 0 if $n < 50$ or if the input is not a byte string at all.▮

4 Enciphering Encoded Messages

Let $\mathsf{Encode} = (\mathit{Encode}, \mathit{Decode})$ be an encoding scheme from $\mathcal{M}$ to $\mathcal{M}^*$ and let $F = \{F_K : \mathcal{M}^* \to \mathcal{M}^*\}$ be a cipher with key space $\mathcal{K}$. Then we define the following encapsulation scheme $F \circ \mathsf{Encode} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$:

(1) $\mathcal{K}$ chooses a random key $K \leftarrow \mathcal{K}$ and outputs it.

(2) $\mathcal{E}_K(M)$ sets $M^* \leftarrow \mathit{Encode}(M)$, returns $\perp$ if $M^* = \perp$, and otherwise computes $C \leftarrow F_K(M^*)$ and returns that. Algorithm $\mathcal{E}$ is stateful if and only if *Encode* is. If *Encode* is stateful then the initial state for $\mathcal{E}$ is the initial state mandated by *Encode*, and $\mathcal{E}$ maintains the state needed by the encoding scheme.

(3) $\mathcal{D}_K(C)$ returns $\perp$ if $C \notin \mathcal{M}^*$, and otherwise computes $M^* \leftarrow F_K^{-1}(C)$, sets $M \leftarrow \mathit{Decode}(M^*)$, and returns M.

For a pictorial representation, see Figure 3.

Privacy from Rare/Collision-Free Encodings. We show that encryption scheme $F \circ \mathsf{Encode}$ is private if encoding scheme Encode has rare or no collisions and F is a secure cipher, in the sense of being a good PRP. The following theorem makes this formal and quantitative.

Theorem 4.1 Let $\mathsf{Encode} = (\mathit{Encode}, \mathit{Decode})$ be an encoding scheme from $\mathcal{M}$ to $\mathcal{M}^*$ and let $F = \{F_K : \mathcal{M}^* \to \mathcal{M}^*\}$ be a cipher with key space $\mathcal{K}$. Suppose that Encode is ϵ-colliding. Then $F \circ \mathsf{Encode} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ has security

$$\mathbf{Adv}^{\mathrm{priv}}_{F\circ\mathsf{Encode}}(t,q,\mu) \quad \leq \quad \mathbf{Adv}^{\mathrm{prf}}_{F}(t',q,\mu) + \epsilon(q)$$

where $t' = t + O(\mu)$.

Proof: Let B be an adversary attacking the privacy of $F \circ \mathsf{Encode}$. Let t be its running time, q the number of queries it makes, and μ the length of all its queries put together, plus the length of B's output. Our goal is to upper bound $\mathbf{Adv}^{\mathrm{priv}}_{F\circ\mathsf{Encode}}(B)$. To this end we introduce a couple of more algorithms and some associated probabilities.

Algorithm D is a distinguisher for F. It is given an oracle for a permutation $f \in \mathbf{Perm}(\mathcal{M}^*)$. It runs B. When B makes an oracle query M, distinguisher D computes $M^* \leftarrow \mathit{Encode}(M)$ and $C \leftarrow f(M^*)$. It returns C to B as the answer to the query. When B terminates, D outputs whatever B outputs.

Algorithm A is a collision finding adversary for Encode. It is given oracle Encode. It picks a permutation f from $\mathbf{Perm}(\mathcal{M}^*)$ at random. (Or simulates such a permutation. The difference is technically immaterial since the running time of A is not restricted.) It then runs B. When B makes an oracle query M, algorithm A computes $M^* \leftarrow \mathit{Encode}(M)$ and $C \leftarrow f(M^*)$. It returns C to B as the answer to the query. When B terminates, so does A.

We now define the following probabilities:

$$\begin{aligned}
p_1 &= \Pr[K \leftarrow \mathcal{K} : B^{\mathcal{E}_K(\cdot)} = 1] \\
p_2 &= \Pr[K \leftarrow \mathcal{K} : B^{\mathcal{E}_K(\$^{|\cdot|})} = 1] \\
p_3 &= \Pr[K \leftarrow \mathcal{K} : D^{F_K(\cdot)} = 1] \\
p_4 &= \Pr[\pi \leftarrow \mathbf{Perm}(\mathcal{M}^*) : D^{\pi(\cdot)} = 1] \\
p_5 &= \Pr[(M_1^*, \ldots, M_q^*) \leftarrow \mathsf{Responses}\ A^{\mathit{Encode}(\cdot)} :\ \exists\, i < j \text{ s.t. } M_i^* = M_j^* \neq \bot] \ .
\end{aligned}$$

Note that $\mathbf{Adv}^{\mathrm{priv}}_{F\circ\mathsf{Encode}}(B) = p_1 - p_2$. To upper bound it we use the following claims.

Claim 1: $p_1 = p_3$.

Proof: This follows from the definitions of D and $F \circ \mathsf{Encode}$. □

Claim 2: $p_2 \geq p_4 - p_5$.

Proof: Let C be the event that there is a collision, meaning $\exists\, i < j$ s.t. $M_i^* = M_j^* \neq \bot$. Let experiment i be that underlying p_i for $i \in \{2,4,5\}$. Event C is defined in all of experiments 2,4 and 5. In all cases, it has the same probability, and conditioned on it not happening, the probability that D outputs 1 in experiment 1 equals the probability that B outputs 1 in experiment 4. Finally, the probability of C is p_5. □ Given these claims we have

$$\mathbf{Adv}^{\mathrm{priv}}_{F\circ\mathsf{Encode}}(B) = p_1 - p_2 \leq p_3 - (p_4 - p_5) = (p_3 - p_4) + p_5 \leq \mathbf{Adv}^{\mathrm{prp}}_{F}(D) + \epsilon(q) \ .$$

This concludes the proof of Theorem 4.1. ▮

Authenticity from Sparse Encodings. We show that $F \circ \mathsf{Encode}$ is an authenticated encryption scheme if encoding Encode adds adequate redundancy and F is a strong PRP. The following theorem makes this formal and quantitative. We remark that this result requires that the PRP be strong, which the previous result did not, and we subsequently show this extra requirement is necessary.

Theorem 4.2 Let $\mathsf{Encode} = (\mathit{Encode}, \mathit{Decode})$ be an encoding scheme from $\mathcal{M}$ to $\mathcal{M}^*$ and let $F = \{F_K : \mathcal{M}^* \to \mathcal{M}^*\}$ be a cipher with key space $\mathcal{K}$. Suppose that Encode is δ-dense and that $q \leq \frac{1}{2\delta}$. Then $F \circ \mathsf{Encode} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ has security

$$\mathbf{Adv}^{\mathrm{auth}}_{F\circ\mathsf{Encode}}(t, q, \mu) \quad \leq \quad \mathbf{Adv}^{\mathrm{sprp}}_{F}(t', q, 2\mu) + 2\delta$$

where $t' = t + O(\mu)$.

Proof: Let B be an adversary attacking the authenticity of $F \circ \mathsf{Encode}$. Let t be its running time, $q-1$ the number of queries it makes, and μ the total length of all its queries put together, and its final output. Our goal is to upper bound $\mathbf{Adv}^{\mathrm{auth}}_{F\circ\mathsf{Encode}}(B)$. To this end we introduce an algorithm D and some probabilities.

Algorithm D is a distinguisher for F. It is given two oracles: f and f^{-1}, where $f \in \mathbf{Perm}(\mathcal{M}^*)$ is a permutation. It runs B. When B makes an oracle query M, distinguisher D computes $M^* \leftarrow \mathit{Encode}(M)$ and $C \leftarrow f(M^*)$. It returns C to B as the answer to the query. When B terminates, it outputs a ciphertext $\mathbf{C}$, which is supposed to its forgery. Algorithm D outputs 0 if $\mathbf{C} \notin \mathcal{M}^*$. Otherwise D computes $\mathbf{M}^* \leftarrow f^{-1}(\mathbf{C})$ (this is the one and only time it uses its f^{-1} oracle). Algorithm D then computes $\mathbf{M} \leftarrow \mathit{Decode}(\mathbf{M}^*)$. If $\mathbf{M} = \perp$ then D outputs 0, else D outputs 1.

We now define the following probabilities:

$$\begin{aligned}
p_1 &= \Pr[K \leftarrow \mathcal{K}\,;\, \mathbf{C} \leftarrow B^{\mathcal{E}_K(\cdot)} : \mathbf{C} \text{ is new and } \mathcal{D}_K(\mathbf{C}) \neq \perp] \\
p_2 &= \Pr[K \leftarrow \mathcal{K} : D^{F_K(\cdot), F_K^{-1}(\cdot)} = 1] \\
p_3 &= \Pr[\pi \leftarrow \mathbf{Perm}(\mathcal{M}^*) : D^{\pi(\cdot), \pi^{-1}(\cdot)} = 1]
\end{aligned}$$

Note that $\mathbf{Adv}^{\mathrm{auth}}_{F\circ\mathsf{Encode}}(B) = p_1$. To upper bound it we use the following claims.

Claim 1: $p_1 = p_2$.

Proof: This follows from the definitions of D and $F \circ \mathsf{Encode}$. □

Claim 2: $p_3 \leq 2\delta$.

Proof: If the ciphertext $\mathbf{C}$ is new and $f = \pi$ is a random permutation, then $\mathbf{M}^* = f^{-1}(\mathbf{C})$ is random subject to not being in the set $\{\, f^{-1}(C_i) \,:\, i = 1, \ldots, q \,\}$ where $C_i = f(\mathit{Encode}(M_i))$ is the response to the i-th query M_i of B. For any integer n let

$$g(n) \;=\; |\{\, M^* \in \{0,1\}^n \;:\; \mathit{Decode}(M^*) \in \{0,1\}^* \,\}| \;.$$

We know $g(n)/2^n \leq \delta$ for all n. In the worst case, all the ciphertexts are of some common length n. The probability that $\mathit{Decode}(\mathbf{M}^*) \neq \perp$ is at most $g(n)/(2^n - q)$. If $g(n) = 0$ then this probability is zero, and hence certainly at most 2δ, so suppose $g(n) \neq 0$. In that case $g(n) \geq 1$, so it must be that $2^n \geq 1/\delta$. But by assumption $q \leq 1/(2\delta)$. So $2^n - q \geq 1/\delta - 1/(2\delta) = 1/(2\delta)$. So $g(n)/(2^n - q) \leq 2\delta$. □ Given these claims we have

$$\mathbf{Adv}^{\mathrm{auth}}_{F\circ\mathsf{Encode}}(B) = p_1 = p_2 = (p_2 - p_3) + p_3 \leq \mathbf{Adv}^{\mathrm{prp}}_{F}(D) + 2\delta \;.$$

This concludes the proof of Theorem 4.2. ▌

We now discuss the necessity of the extra requirement on the PRP above, namely that it be strong. The following indicates that without this requirement, the authenticity does not hold. Using the bounds found in the proof, the informal theorem statement below is easily adapted to give a more precise (but less understandable) quantitative assertion.

Theorem 4.3 If there exists a secure PRP then there exists a secure PRP F (that is not a strong-PRP) and a δ-dense encoding scheme Encode for which the scheme $F \circ \mathsf{Encode}$ does not achieve authenticity.▮

Proof: Let $\mathcal{M} = \mathcal{K} = \{0,1\}^n$ and $\mathcal{M}^* = \{0,1\}^{2n}$. Let $\mathsf{Encode} = (\mathit{Encode}, \mathit{Decode})$ be the encoding scheme from $\mathcal{M}$ to $\mathcal{M}^*$ in which $\mathit{Encode}(M) = M0^n$, namely the encoding function appends n zeros to the input. The *Decode* function on input a $2n$ bit string M^* outputs $\perp$ if the last n bits of M^* are not all zero, and otherwise outputs the last n bits of M^*. This encoding scheme is δ-dense where $\delta = 2^{-n}$.

Assume we are given a secure PRP $G = \{G_K : \mathcal{M}^* \to \mathcal{M}^*\}$ with key space $\mathcal{K}$. We construct a secure PRP $F = \{F_{K_1,K_2} : \mathcal{M}^* \to \mathcal{M}^*\}$ with key space $\mathcal{K} \times \mathcal{K}$. This will have the property that $F \circ \mathsf{Encode} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ can be broken from the authenticity point of view.

We will design F so that for any key K we have $F_{K_1,K_2}(K_2 0^n) = 0^{2n}$. Let us first see why this furnishes the necessary example, and then see how to construct F from G.

The attack on $F \circ \mathsf{Encode}$ is as follows. The adversary makes no queries. It simply outputs $C = 0^{2n}$. Now notice that $\mathit{Decode}(F^{-1}_{K_1,K_2}(C)) = \mathit{Decode}(K_2 0^n) = K_2$ is not equal to $\perp$, so the adversary wins, with probability one. This shows the scheme is totally insecure.

Notice that F is definitely *not* a strong-PRP. If a distinguisher has oracles f, f^{-1}, it can call f^{-1} on 0^{2n}, and obtain a reply xy. It returns 1 if $y = 0^n$ and 0 otherwise. The probability it returns 1 is 1 if $f = F_{K_1,K_2}$, and 2^{-n} if f is a random permutation on $\mathcal{M}^*$, so its advantage is almost one.

To complete the proof we need to show how to construct F from G so that it preserves the PRP-ness and invertibility of G–

```
**Algorithm** $F_{K_1,K_2}(xy)$ (Here $|x| = |y| = |K_1| = |K_2| = n$)
    **If** $xy = K_2 0^n$ **then return** $0^{2n}$
    **Else If** $G_{K_1}(xy) = 0^{2n}$ **then return** $G_{K_1}(K_2 0^n)$
        **Else return** $G_{K_1}(xy)$
```

We need to show that F is a family of permutations (meaning each F_{K_1,K_2} is invertible) and also that F is almost as secure as G in the PRP sense. To see the first, note that the following works as the inverse function–

$$F^{-1}_{K_1,K_2}(wz) \;=\; \begin{cases} K_2 0^n & \text{if } wz = 0^{2n} \\ G^{-1}_{K_1}(0^{2n}) & \text{if } wz = G_{K_1}(K_2 0^n) \\ G^{-1}_{K_1}(wz) & \text{otherwise.} \end{cases}$$

That F is still a secure PRP is because an adversary given an oracle f is unlikely to be able to touch f in any of the "bad" places, since one of them would correspond to knowing a part of the key, and another to inverting a random permutation on a fixed point. More precisely we claim that for any $q \le 2^n/4$ and any t, μ we have

$$\mathbf{Adv}^{\mathrm{prp}}_F(t,q,\mu) \le \mathbf{Adv}^{\mathrm{prp}}_G(t+O(n),q,\mu) + \frac{4q}{2^n} \,. \qquad (1)$$

To see this, let D_F be a distinguisher for F that makes q queries, runs in time t, and queries a total of μ bits. We define a distinguisher D_G for G. The code is below and explanations follow–

```
**Algorithm** $D_G^g$ (Here $g$: $\mathcal{M}^* \to \mathcal{M}^*$)
    $K_2 \leftarrow \{0,1\}^n$
    **For** $i = 1, \ldots, q$ **do**
        $D_F \to x_i y_i$
        $w_i z_i \leftarrow g(x_i y_i)$
        **If** $x_i = K_2$ and $y_i = 0^n$ **then return** 1
        **Else If** $w_i z_i = 0^{2n}$ **then return** 1
        **Else** $D_F \leftarrow w_i z_i$
    $D_F \leftarrow b$
    **return** $b$
```

Distinguisher D_G first picks a string K_2 at random to play the role of the second part of the key for F. It then starts running D_F. The notation $D_F \to x_i y_i$ means D_F makes query $x_i y_i$. D_G must answer it. It answers by the value of g on this point, except for two special cases, in each of which D_G halts and returns 1. If D_G did not halt in the query process it obtains the final decision bit b of D_F, and returns the same.

We claim that

$$\Pr[K \leftarrow \mathcal{K} : D_G^{G_K(\cdot)} = 1] \;\; \geq \;\; \Pr[(K_1, K_2) \leftarrow \mathcal{K} \times \mathcal{K} : D_F^{F_{K_1,K_2}(\cdot)} = 1] \tag{2}$$

$$\Pr[\pi \leftarrow \mathbf{Perm}(\mathcal{M}^*) : D_G^{\pi(\cdot)} = 1] \;\; \leq \;\; \Pr[\pi \leftarrow \mathbf{Perm}(\mathcal{M}^*) : D_F^{\pi(\cdot)} = 1] + \frac{2q}{2^n} \,. \tag{3}$$

We will argue this below. Assuming it, subtraction gives us

$$\mathbf{Adv}_F^{\mathrm{prp}}(D_F) \leq \mathbf{Adv}_G^{\mathrm{prp}}(D_G) + \frac{4q}{2^n} \,,$$

which implies Equation (1).

Equation (2) is true because D_G simulates D_F except in some cases, but in those cases it returns 1, so its probability of returning 1 can only increase with respect to that of D_F. On the other hand, the difference

$$\Pr[\pi \leftarrow \mathbf{Perm}(\mathcal{M}^*) : D_G^{\pi(\cdot)} = 1] - \Pr[\pi \leftarrow \mathbf{Perm}(\mathcal{M}^*) : D_F^{\pi(\cdot)} = 1]$$

is bounded above by the probability that one of the special cases occurs when running D_G with a random permutation π. Each time, two of the remaining inputs could cause one of the special cases to happen, so the chance is bounded above by

$$\sum_{i=0}^{q-1} \frac{2}{2^n - 2i} \;\; \leq \;\; \frac{2q}{2^n/2} \,,$$

the last inequality exploiting the assumption that $q \leq 2^n/4$. This gives us Equation (3) and completes the proof. ▌

Acknowledgments

Mihir Bellare was supported in part by NSF CAREER AWARD CCR-9624439 and a Packard Foundation Fellowship in Science and Engineering. Phillip Rogaway was supported in part under NSF CAREER Award CCR-962540, and under MICRO grants 97-150 and 98-129, funded by RSA Data Security, Inc..

References

[1] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, "A concrete security treatment of symmetric encryption." *Proceedings of the 38th Symposium on Foundations of Computer Science*, 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, pp. 362–399.

[3] M. Bellare, T. Krovetz and P. Rogaway, "Luby-Rackoff backwards: Increasing security by making block ciphers non-invertible." *Advances in Cryptology – EUROCRYPT '98*, Lecture Notes in Computer Science Vol. 1403, K. Nyberg ed., Springer-Verlag, 1998.

[4] 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 Vol. 1976, T. Okamoto ed., Springer-Verlag, 2000.

[5] M. Bellare and P. Rogaway, "On the construction of variable-input-length ciphers." *Fast Software Encryption '99*, Lecture Notes in Computer Science Vol. 1636, L. Knudsen ed., Springer-Verlag, 1999.

Advances in Cryptology – ASIACRYPT '00, Lecture Notes in Computer Science Vol. 1976, T. Okamoto ed., Springer-Verlag, 2000.

[6] D. Dolev, C. Dwork and M. Naor. "Non-malleable cryptography," *Proceedings of the 23rd Annual Symposium on the Theory of Computing*, ACM, 1991. To appear in *SIAM J. on Computing.*

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

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

[9] J. Katz and M. Yung, "Unforgeable encryption and adaptively secure modes of operation." *Fast Software Encryption '00*, Lecture Notes in Computer Science, B. Schneier, ed., Springer-Verlag, 2000.

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

[11] M. Naor and O. Reingold, "On the construction of pseudo-random permutations: Luby-Rackoff revisited." *J. of Cryptology*, vol. 12, 1999, pp. 29–66.

[12] C. Rackoff and D. Simon, "Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack." *Advances in Cryptology – CRYPTO '91*, Lecture Notes in Computer Science Vol. 576, J. Feigenbaum ed., Springer-Verlag, 1991.

[13] R. Rivest, "All-or-nothing encryption and the package transform." *Fast Software Encryption '97*, Lecture Notes in Computer Science Vol. 1267, E. Biham ed., Springer-Verlag, 1997.

[14] C. Shannon, "Communication theory of secrecy systems." Bell Systems Technical Journal, 28(4), 656–715 (1949).