

On the Construction of Variable-Input-Length Ciphers

MIHIR BELLARE* GIOVANNI DI CRESCENZO† PHILLIP ROGAWAY‡

December 14, 1998

Abstract

We investigate how to construct ciphers which operate on messages of various (and effectively arbitrary) lengths. In particular, lengths *not necessarily* a multiple of some block length. (By a “cipher” we mean a key-indexed family of length-preserving permutations, with a “good” cipher being one that resembles a family of random length-preserving permutations.) Oddly enough, this question seems not to have been investigated. We show how to construct variable-input-length ciphers starting from any block cipher (ie, a cipher which operates on strings of some fixed length n). We do this by giving a general method starting from a particular kind of pseudorandom function and a particular kind of encryption scheme, and then we give example ways to realize these tools from a block cipher. All of our constructions are proven sound, in the provable-security sense of contemporary cryptography. Variable-input-length ciphers can be used to encrypt in the presence of the constraint that the ciphertext be no longer than the plaintext, and they may prove to be generally useful cryptographic tools.

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

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

†Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. E-Mail: giovanni@cs.ucsd.edu.

‡ Dept. of Computer Science, Engineering II Bldg., One Shields Ave., University of California at Davis, Davis, CA 95616, USA. E-Mail: rogaway@cs.ucdavis.edu URL: <http://www.cs.ucdavis.edu/~rogaway/>

1 Introduction

The construction and analysis of block ciphers (continuing now with the AES) is complemented by another line of work: that which seeks to better understand the use of block ciphers in the design of higher-level primitives. The latter can serve to guide the former. For example, attacks and proofs of security which identified the birthday threshold as the security bottleneck for the CBC MAC and for CBC encryption [3, 2, 17] have lead designers to increase the block length (not just the key length) of new ciphers. Similarly, the AES has the explicit design requirement that the block cipher be “pseudorandom,” to some extent influenced by the proofs of security for various block-cipher based constructions under this assumption [3, 2].

This paper introduces the question of how to construct ciphers which act on variable-length inputs. Starting from a block cipher, we provide a class of provably-good solutions to this problem. These schemes can be easily and efficiently instantiated. We believe that they cipher will prove to be useful and general cryptographic tools.

WHAT IS A VARIABLE-INPUT-LENGTH CIPHER? A family of permutations $F = \{F_K\}$ will henceforth be called a *cipher*. Applying one of these functions F_K will be called *enciphering*. Applying F_K^{-1} will be called *deciphering*. In this paper, “good” for an enciphering method is understood to mean approximating a family of random permutations.

Block ciphers operate on inputs that have a fixed length, the block length, which we denote by n . There are various constructions that will extend a given block cipher to a cipher on inputs of length ℓn for certain integers $\ell \geq 1$. These are the constructions of “pseudo-random permutations” [8, 9, 16, 14, 20, 11]. (They require ℓ to be even.) If we need to encipher an input of length not a multiple of n , the only way to do it would be to pad the input and then apply the cipher, resulting in a ciphertext that is longer than the plaintext.

We would like a cipher that can take an input of any length (in particular not a multiple of some block length) and return an output of the same length as the input. This is a variable-input-length cipher.

In other words, presumably we’ll be seeing a number of messages, M_1, M_2, \dots , and these messages may have varying and effectively arbitrary lengths. We want to create the ciphertexts C_1, C_2, \dots . We have a block cipher available, but we certainly don’t have a block cipher available (and all separately keyed!) for each possible input length. So we want a variable-input-length cipher, which uses just one key, but can still be applied to an input of any length to yield an output of the same length as the input.

WHY DO WE WANT VARIABLE-INPUT-LENGTH CIPHERS? An important attribute in some applications is to be able to do “length-preserving encryption,” meaning obtain “ciphertexts” that are the same length as plaintexts. Pseudo-random permutations have been suggested as a solution, but as long as they are restricted to input lengths multiple of some block length, don’t really provide this function. What we need is a variable-input-length cipher.

Is it really worth the trouble to save a few bytes of ciphertext? Yes, often enough that the problem is worth addressing. In networking applications, for example, a “packet format” may have been defined, where this packet format describes all sorts of fields, none of which were intended for cryptographic purposes. Now suppose a need arises to add in privacy features. At the same time, there will often be a real-world constraint making it is no longer desirable (or even feasible) to grow or re-define the packet format. Enciphering with a variable-input-length cipher leaves the packet

size alone, and packets are left looking identical (after deciphering) to the way they looked before.

ENCIPHERING VS. ENCRYPTING. What kind of privacy or other security properties can enciphering provide? This needs a careful look. One must begin by clarifying the relationship between the (often confused) operations of enciphering and encrypting. Many popular books on cryptography describe the application of a key-indexed permutation F_K to the plaintext M (what we called enciphering) as “encrypting.” Yet our community has long recognized that encryption realized this way, being deterministic, cannot possibly achieve the strong privacy guarantees (semantic security and beyond) that one would hope for [6, 2]. (For example, if the same message is enciphered twice, an adversary will detect this.)

The advantage of enciphering over encrypting is a shorter ciphertext: secure encryption is inherently length-increasing, while secure enciphering can be (and in this paper, always will be) length-preserving. But we must give up a little in the kind of privacy obtained. What kind of privacy does enciphering provide?

There seems to be a widespread belief that enciphering a message is, somehow, *almost* as good as encrypting it. This intuition can be formalized along the following lines. With a good encryption scheme, all that an adversary can learn about the plaintext, given the ciphertext, is the length of the plaintext. With a good cipher, all that an adversary can learn about the plaintext, given the ciphertext, is the length of the plaintext *and which earlier plaintext, if any, this ciphertext is again the encipherment of*. This additional increment of leaked information is all that “disqualifies” good enciphering from being good encrypting.

In many instances, leaking this particular piece of information is simply irrelevant. When it is known *a priori*, for example, that messages contain a sequence number, then leaking which messages are identical to which others is leaking nothing at all; the adversary already *knows* that all messages are distinct.

We comment that if one of the fields of a packet functions as a nonce or sequence number, then enciphering the packet will already provide for semantic security, and in a more robust and generic way than trying to directly exploit that field cryptographically.

MAKING TOOLS. We have just argued that, depending on the application domain, enciphering may be a good-enough substitute for encrypting. The length-saving may justify some extra work. But a further reason to investigate this question is that variable-input-length ciphers may prove to be useful tools for protocol design. Obviously block ciphers, even with their length-restriction, have proven to be highly versatile design tools. Variable-input-length ciphers should be versatile, too.

CONSTRUCTING VARIABLE-INPUT-LENGTH CIPHERS. We want to be able 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 $\pi : \mathcal{M} \rightarrow \mathcal{M}$. We want to build it starting from a given block cipher E that has block size n . The difficulty in this construction is two-fold: making sure our constructed cipher is both length-preserving and invertible.

We suggest a simple and efficient approach for making such variable-input-length ciphers out of block ciphers. Here we illustrate the idea with a concrete example. Two passes are made over the cipher’s input M . In the first pass we compute what is essentially the CBC MAC of M . Actually an extension of the CBC MAC must be used—such as re-enciphering the final ciphertext block using a different key [15]—in order to correctly handle messages of varying input lengths. This gives as an n -bit MAC. Now use this MAC as the initial value of a “counter” to encrypt in counter

mode a prefix of M consisting of all but the last n bits of M . (Note this encryption does not require the length of the plaintext to be a multiple of the block length.) The ciphertext of M is the MAC computed during the CBC-MAC pass concatenated to the ciphertext computed during the encryption pass. Different keys are used during the MAC computation and the CBC encryption. The thing that makes this a cipher (ie, invertible) is the following observation about the CBC MAC: that if you know the CBC MAC of a message M and the underlying MAC key, and you also know all of the blocks of M except for the last one, then you can recover the last message block, too. A more general exposition of this approach is given in Section 3.

Note that the cost of our enciphering was twice the customary cost of encrypting. We don't know if this is essential. It obviously is essential, however, to examine all message bits before any ciphertext bit is produced.

RELATED WORK. There is work on constructing block ciphers of one blocksize given block ciphers of another blocksize, but to the best of our knowledge, not on making ciphers that can handle input lengths which vary.

For example, Luby and Rackoff [8] consider the question of how to turn an n -bit to n -bit pseudorandom function (PRF) into a $2n$ -bit to $2n$ -bit block cipher. They show that 3 rounds of the Feistel construction suffices for this purpose, and that 4 rounds suffice to obtain a “super” PRP from a PRF. The paper has spawned much work, with [9, 16, 14, 20] to name a few. Naor and Reingold [11] provide a construction which extends an n -bit block cipher to a block cipher of $N = 2ni$ bits, for any desired $i \geq 0$. It is again unclear how to use their construction for arbitrary N and across assorted lengths.

Rivest puts forward the idea of “all-or-nothing” encryption [18], wherein an adversary, guessing the key, should have to invest $\Theta(|C|)$ time before obtaining information useful to verify her guess. Enciphering the message as we do here would achieve the this goal.

2 Definitions

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 . We assume the existence of an (efficient) algorithm to decide membership in \mathcal{M} . 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.

When we speak of the running time of an algorithm by convention we mean the actual running time plus the size of the length of the description of that algorithm.

CIPHERS AND PRFs. Let \mathcal{K} , \mathcal{M} and \mathcal{C} be a key space, message space, and ciphertext space. A *pseudorandom function* (PRF) is a collection of functions $F = \{F_K \mid K \in \mathcal{K}\}$, each $F_K : \mathcal{M} \rightarrow \mathcal{C}$ sharing the same domain \mathcal{M} and range \mathcal{C} . We assume that $|F_K(M)| = \ell(|M|)$ depends only on $|M|$. We call ℓ the *length function* of the PRF.

A *cipher* is a PRF $F = \{F_K \mid K \in \mathcal{K}\}$ in which each $F_K : \mathcal{M} \rightarrow \mathcal{C}$ 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*.

Let \mathcal{M} be a message space and let ℓ be a length function. We define two “reference” PRFs:

$\text{Rand}(\mathcal{M}, \ell)$ A random function ρ from this set is determined as follows: for each $M \in \mathcal{M}$ let $\rho(M)$ be a random string in $\{0, 1\}^{\ell(|M|)}$.

$\text{Perm}(\mathcal{M})$ A random function π from this set 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 $i = |M|$.

Thus $\rho \leftarrow \text{Rand}(\cdot, \cdot)$ is used to choose a random function and $\pi \leftarrow \text{Perm}(\cdot)$ is used to choose random permutation. The arguments are used to indicate the domain and range you desire.

SECURITY OF PRFS AND CIPHERS. We follow the formalization of [5], adapted to concrete security as in [3]. A *distinguisher* is a (possibly probabilistic) algorithm A with access to an oracle \mathcal{O} . Let A be a distinguisher and let $F = \{F_K \mid K \in \mathcal{K}\}$ be a PRF with key space \mathcal{K} and length function ℓ . Then we let

$$\text{Adv}_A^{\text{prf}}(F) = \Pr[K \leftarrow \mathcal{K}; A^{F_K(\cdot)} = 1] - \Pr[\rho \leftarrow \text{Rand}(\mathcal{M}, \ell); A^{\rho(\cdot)} = 1]$$

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

$$\text{Adv}_A^{\text{prp}}(F) = \Pr[K \leftarrow \mathcal{K}; A^{F_K(\cdot)} = 1] - \Pr[\pi \leftarrow \text{Perm}(\mathcal{M}); A^{\pi(\cdot)} = 1]$$

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

$$\text{Sec}_F^{\text{prf}}(t, q, \mu) = \max_A \{\text{Adv}_A^{\text{prf}}(F)\} \quad \text{and} \quad \text{Sec}_F^{\text{prp}}(t, q, \mu) = \max_A \{\text{Adv}_A^{\text{prp}}(F)\}$$

where the maximum is taken over all adversaries which run in time at most t and ask 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 $-$. 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.

ENCRYPTION SCHEMES. Fix a key space \mathcal{K} , message space \mathcal{M} , and ciphertext space \mathcal{C} . An *encryption scheme* is a triple of algorithms $\Pi = (\text{Keygen}, \text{Encrypt}, \text{Decrypt})$ as we now describe.

The key generation algorithm Keygen is a probabilistic algorithm that produces a key $K \in \mathcal{K}$.

Algorithm Encrypt can be either probabilistic or stateful. If probabilistic it takes a key $K \in \mathcal{K}$ and a message $M \in \{0, 1\}^*$, and it flips some coins $r \in \{0, 1\}^*$. The algorithm then returns $C = \text{Encrypt}(M; r)$. If stateful the algorithm takes a key $K \in \mathcal{K}$ and a message $M \in \{0, 1\}^*$, and it uses its internal state $r \in \{0, 1\}^*$. The algorithm then returns $C = \text{Encrypt}(M; r)$, and it may modify its internal state to some new state, r' . Either way, C can be either a binary string in \mathcal{C} or the distinguished symbol $-$. The value $-$ is used 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 Decrypt takes $K \in \mathcal{K}$ and $C \in \{0, 1\}^*$ and computes $M = \text{Decrypt}(C)$ where M is either a string in \mathcal{M} or the distinguished symbol $-$. A return value of $-$ is used to indicate that C is regarded as inauthentic. We call C *valid* if $\text{Decrypt}_K(C) \in \mathcal{M}$ and we call C *invalid* if $\text{Decrypt}_K(C) = -$. We also permit applying Encrypt to $(-, r)$, which results in a return value of $-$. Likewise, applying Decrypt_K to $-$ is permitted and this gives a return value of $-$.

We require the following: if $C = \text{Encrypt}(M; r)$ and $C \neq -$ then $\text{Decrypt}_K(C) = M$.

PRIVACY. Several formalizations for the security of a symmetric encryption scheme under chosen-plaintext attack were provided in [2] and compared in terms of concrete security. We will use one of their 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 $S = (\text{Keygen}, \text{Encrypt}, \text{Decrypt})$ 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

$$\text{Adv}_A^{\text{priv}}(\Pi) = \Pr [K \leftarrow \text{Keygen} : A^{\text{Encrypt}_K(\cdot)} = 1] - \Pr [K \leftarrow \text{Keygen} : A^{\text{Encrypt}_K(\mathcal{S}^{\lfloor \cdot \rfloor})} = 1] .$$

In the first game, the oracle, given a message, returns its encryption under key K ; in the second game the oracle, given a message, ignores it except to record its length n , and then returns the 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 define

$$\text{Sec}_{\Pi}^{\text{priv}}(t, q, \mu) = \max_A \{ \text{Adv}_A^{\text{priv}}(\Pi) \}$$

where the maximum is over all adversaries who run in time at most t and ask at most q oracle queries, where these queries total at most μ bits.

3 Constructing Variable-Input-Length Ciphers

To encrypt messages of arbitrary (and varying) length using the encode-then-encipher approach one needs first a cipher F which will, under a key K , encipher messages of arbitrary (and varying) lengths. Though this is a seemingly “basic” object, neither the literature nor prevailing practice seems to provide any guidance on how to construct such ciphers. Note, in particular, that block ciphers only act on messages of some fixed length (the blocksize); and constructions such as the Feistel construction only provide a way to go from a block cipher of one fixed size to a block cipher of another fixed size.

To gain some intuition, here are a couple of ideas which do *not* work. Assume we have in hand some block cipher E which operates on n -bit blocks. We are trying to construct a cipher F with some enlarged domain — say messages of *one or two* n -bit blocks. Remember that for F to be “good” $F_K(\cdot)$ should look like a random length-preserving permutation (when K is random and unknown).

- One might just try using the CBC-MAC with a zero initialization vector: $C_i = E_K(C_{i-1} \oplus M_i)$, where $C_0 = 0^n$. But this is no good: to distinguish this construction from a random permutation just ask queries $M_1 M_2 = 0^n 0^n$ and $M'_1 M'_2 = 0^n 1^n$. These result in enciphered strings $C_1 C_2$ and $C'_1 C'_2$ ($|C_1| = |C'_1| = n$) where $C_1 = C'_1$ under the suggested construction even though this equality hardly ever holds with a random permutation.
- What about applying the underlying block cipher E when the message is one block and using a three-round Feistel network (using underlying block cipher) when the message has two blocks. Again this does not work. From the successive encipherments of three one-block messages one can figure out what of some (unqueried) two-block message.

The above examples may make clear that a practical method for making a variable-input-length cipher is not very obvious.

The construction we suggest uses two tools: a particular kind of pseudorandom function and a particular kind of symmetric encryption scheme. We call these tools a *parsimonious* PRF and a *parsimonious* encryption scheme. Efficient constructions for these objects are easily obtained from

a block cipher, and we will show how to do this. We begin by describing the construction with a concrete example.

3.1 A Concrete Example

We first describe a concrete instantiation of our scheme. But first we will need to develop a new variant of the CBC MAC.

EXTENDED CBC MAC. Let us begin by reviewing the “basic” CBC MAC of a block cipher E having keyspace \mathcal{K} and blocksize n . We are given a message $M_1 \cdots M_\ell$ with $\ell \geq 1$ blocks, $|M_1| = \cdots = |M_\ell| = n$. Then $\text{CBC-MAC}_K^E(M)$ is defined as C_ℓ where $C_0 = 0^n$ and $C_i = E_K(M_i \oplus C_{i-1})$ for $1 \leq i \leq \ell$.

The CBC MAC has an interesting if simple property which we have not seen mentioned before. The property is this. Let $M = M_1 \cdots M_\ell$ be a message of $\ell \geq 1$ blocks, each block being n -bits long. Suppose you know the key K and the MAC $\sigma = \text{CBC-MAC}_K^E(M)$ of the message M under key K . Suppose you also know $M_{\text{prefix}} = M_1 \cdots M_{\ell-1}$ — all of the message M except for its last n bits. Then you can compute these last n bits, M_ℓ . To do this, first compute $C_0, C_1, \dots, C_{\ell-1}$, where $C_0 = 0^n$ and $C_i = E_K(M_i \oplus C_{i-1})$ for $1 \leq i \leq \ell - 1$. Then, since $\sigma = E_K(M_\ell \oplus C_{\ell-1})$, you can compute M_ℓ as $C_{\ell-1} \oplus E_K^{-1}(\sigma)$. We say that the basic CBC MAC is *parsimonious*.

If E is a good block cipher then $G = \text{CBC-MAC}^E$ is a good pseudorandom function, as demonstrated by [3]. But remember this is so only with respect to inputs of some fixed size. We will need a PRF that works across input of any length. The CBC MAC can be generalized to work for more than fixed-length strings. Methods are described in various standards, such as [7]. One such generalization is provided and proven correct in [15]. It works across strings of an arbitrary number of blocks, What we would like is a bit more still: a good PRF across all message lengths $\geq n$ bits, while still preserving the parsimoniousness property which we have just described. Here is how we do this.

We are starting with E , an n -bit block cipher with key space \mathcal{K} . Let $K, K' \in \mathcal{K}$ and let $M \in \{0, 1\}^{\geq n}$. Then define $\text{X-CBC-MAC}_{KK'}^E(M)$ as the value returned by the following algorithm.

Algorithm $\text{X-CBC-MAC}_{KK'}^E(M)$

- (1) Let pad be a “1” followed by the minimum number of “0” bits such that $|M| + |pad|$ is divisible by n . Insert pad into M just before the last ℓ bits of M : if $M = M_{\text{prefix}} \parallel M_{\text{suffix}}$, where $|M_{\text{suffix}}| = n$, then replace M by $M_{\text{prefix}} \parallel pad \parallel M_{\text{suffix}}$.
- (2) Partition M into n -bit blocks: $M = M_1 \cdots M_{\ell-1} M_\ell$.
- (3) Let $C_0 = 0^n$, and let $C_i = E_K(C_{i-1} \oplus M_i)$ for all $1 \leq i \leq \ell - 1$.
- (4) Return $C_\ell = E_{K'}(C_{\ell-1})$.

An application of Lemma 3.1 from [15] can be used to establish the following theorem. It is a quantitative result on the efficacy of X-CBC-MAC in terms of the efficacy of E . Details are omitted.

Theorem 3.1 Let $E : \mathcal{K} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a block cipher and Let $G = \text{X-CBC-MAC}^E$. Then

$$\text{Sec}_G^{\text{prf}}(t, q, \mu) \leq 2 \cdot \text{Sec}_E^{\text{prf}}(t', q') + (\mu + nq)^2 2^{-n}$$

where $t' = t + O(\mu + nq)$ and $q' = \lfloor (\mu + nq)/n \rfloor$, provided that $(\mu + nq)^2 \leq 2^{(n+1)/2}$.

THE CONSTRUCTION IS PARSIMONIOUS. The following algorithm demonstrates that X-CBC-MAC is parsimonious. Let $K, K' \in \mathcal{K}$, $M = \{0, 1\}^{\geq n}$, $M \in \mathcal{M}$, and $\sigma = \text{X-CBC-MAC}_{KK'}^E(M)$. Let M_{prefix} be the first $|M| - n$ bits of M . Then we can recover M_{suffix} , the remaining n bits of M , as follows:

Algorithm Recover (KK' , M_{prefix} , σ)

- (1) Append to M_{prefix} a “1” bit and then the minimal number of “0” bits such that the length of the padded M_{prefix} is divisible by n .
- (2) Partition M_{prefix} into n -bit blocks: $M_{\text{prefix}} = M_1 \cdots M_{\ell-1}$.
- (3) Let $C_0 = 0^n$, and let $C_i = E_K(C_{i-1} \oplus M_i)$ for all $1 \leq i \leq \ell - 1$.
- (4) Return $M_{\text{suffix}} = C_{\ell-1} \oplus E_{K'}^{-1}(\sigma)$.

It is easy to check that the algorithm “works” to recover M_{suffix} .

MAKING A VARIABLE-INPUT-LENGTH CIPHER. Given a block cipher E with key space \mathcal{K} and blocksize n , we can encipher a message $M \in \{0, 1\}^{\geq n}$ under a key $KK'K'' \in \mathcal{K}^3$ as follows:

Algorithm Encipher $_{KK'K''}^E(M)$

- (1) Let $\sigma = \text{X-CBC-MAC}_{KK'}^E(M)$.
- (2) Let M_{prefix} be the first $|M| - n$ bits of M .
- (3) Let pad be the first $|M| - n$ bits of $E_{K''}(\sigma) \parallel E_{K''}(\sigma + 1) \parallel E_{K''}(\sigma + 2) \cdots$.
- (4) Let $C_{\text{prefix}} = M_{\text{prefix}} \oplus pad$.
- (5) Return ciphertext $C = \sigma \parallel C_{\text{prefix}}$.

Naturally $\sigma + i$, occurring as an argument to the block cipher, means to regard σ as a number in the conventional way, add i modulo 2^n , and then regard the resulting number again as an n -bit string in the conventional way.

To decipher ciphertext $C \in \{0, 1\}^{\geq n}$ using key $KK'K''$ one does the following:

Algorithm Decipher $_{KK'K''}^E(C)$

- (1) Partition C into its first n bits, σ , and its remaining bits, C_{prefix} .
- (2) Let pad be the first $|C_{\text{prefix}}|$ bits of $E_{K''}(\sigma) \parallel E_{K''}(\sigma + 1) \parallel E_{K''}(\sigma + 2) \cdots$.
- (3) Let $M_{\text{prefix}} = pad \oplus C_{\text{prefix}}$.
- (4) Let $M_{\text{suffix}} = \text{Recover}(KK', M_{\text{prefix}}, \sigma)$.
- (5) Return plaintext $M = M_{\text{prefix}} \parallel M_{\text{suffix}}$

3.2 Parsimonious PRFs and Encryption Schemes

Towards the generalization and proof we now make some definitions.

DEFINITION OF A PARSIMONIOUS PRF. Let \mathcal{K} , \mathcal{M} and \mathcal{C} be the key space, domain, and range for a pseudorandom function $G = \{G_K : K \in \mathcal{K}\}$. Assume that the length function for G is the constant n ; that is, $|G_K(M)| = n$ for all messages $M \in \mathcal{M}$. To keep things simple we further assume that \mathcal{M} includes only strings of length at least n . Such a PRF G is said to be *parsimonious* if for all $K \in \mathcal{K}$ and all $M \in \mathcal{M}$, the last n bits of M are uniquely determined by the remaining bits of M , the key K , and $G_K(M)$.

In other words, with a parsimonious PRF G , if you know K and receive the n -bit value $G_K(M)$, then you don't need to receive all of M in order to know what it is: it is sufficient to get its first $|M| - n$ bits. The remaining n bits can be recovered using the information already in hand.

EXAMPLE. We have already given an example of a parsimonious PRF, namely X-CBC-MAC,

DEFINITION OF A PARSIMONIOUS ENCRYPTION SCHEME. Let $S = (\text{Keygen}, \text{Encrypt}, \text{Decrypt})$ be a probabilistic symmetric encryption scheme of the following special form: algorithm Encrypt randomly chooses a string $IV \leftarrow \{0, 1\}^n$, and then the ciphertext of M is a $C = IV \parallel C^*$, where $|C^*| = |M|$. We call such an encryption scheme *parsimonious*.

Common modes of encryption using a block cipher, like counter mode or CBC-mode with a random IV, are all parsimonious. In fact, the only natural way to construct from a block cipher a secure but *not* parsimonious encryption scheme is to allow messages whose length is not a multiple of the input length, and to handle these odd-size messages in a way that grows the ciphertext by the length of IV plus the length of the padding. In fact, there are always ready alternatives to such methods.

EXAMPLE 1. We have already given one example of a parsimonious encryption scheme, in the form of counter-mode encryption. Here one encrypts M under key K by

$$IV \parallel (M \oplus (E_K(IV) \parallel (E(IV + 1) \parallel E(IV + 2) \parallel \dots))),$$

where $IV \leftarrow \{0, 1\}^n$ is selected at random, addition is understood as in our earlier example, and the indicated XOR means to XOR M with the first $|M|$ bits of $E_K(IV) \parallel E(IV + 1) \parallel E(IV + 2) \parallel \dots$. Bounds on the security for this scheme are provided in [2].

EXAMPLE 2. The most prevalent encryption scheme is probably CBC mode with a random IV, where the message space is $\mathcal{M} = (\{0, 1\}^n)^+$ and one encrypts the ℓ -block message $M_1 \cdots M_\ell$ by $IV \parallel C_1 \cdots C_\ell$, where $C_i = E_K(C_{i-1} \oplus M_i)$ for $1 \leq i \leq \ell$ and $IV = C_0 \leftarrow \{0, 1\}^n$ is selected at random. This scheme is parsimonious. Bounds on its security are provided in [2].

EXAMPLE 3. CBC mode encryption can readily be extended to parsimonious encryption on domain $\mathcal{M} = \{0, 1\}^*$ using the method, for example, of IBM CUSP/3848. The method is identical to CBC encryption as described above when the message is a multiple of n bits, but when it is not the last (short) block is enciphered by XORing it with the corresponding leftmost bits of the result of applying E_K to the previous (full) block of ciphertext.

3.3 General Construction

We now describe a generalization of the construction in Section 3.1. This is what we will prove secure. The security of the example construction follows.

Let G be a parsimonious PRF with key space \mathcal{K}_{prf} , domain \mathcal{M} , and length function n . Let $S = (\text{Keygen}, \text{Encrypt}, \text{Decrypt})$ be a parsimonious encryption scheme whose message space includes all strings whose length is n less than the length of a string in \mathcal{M} . Let \mathcal{K}_{enc} be the distribution on keys induced by Keygen . Then we define from G and S the following cipher F : the keyspace is $\mathcal{K} = \mathcal{K}_{\text{prf}} \times \mathcal{K}_{\text{enc}}$; the domain is \mathcal{M} ; and, letting $K = (K_{\text{prf}}, K_{\text{enc}})$ be a key, the ciphertext C of $M \in \mathcal{M}$ is

$$C = F_K(M) = \text{Encrypt}_{K_{\text{enc}}}(M_{\text{prefix}}; G_{K_{\text{prf}}}(M)),$$

where $M_{\text{prefix}} = M[1 \dots |M| - n]$ is M stripped of its last n bits. In other words, you apply the PRF to M to compute IV; you encrypt M_{prefix} using IV; and you output that ciphertext (which, recall, begins with IV). The key used for the PRF and the key used for encryption are chosen independently.

In our earlier example, G was X-CBC-MAC^E and the encryption scheme S was counter mode using E .

DROPPING THE $|M| \geq n$ RESTRICTION. We have focussed on the case in which the message length is at least the block length n (and that is what we will analyze in the next section). However, we point out that our method is easily extended to allow for enciphering messages whose length is less than the block length.

For shorter messages of even length, proceed as follow. First map the underlying enciphering key K into subkeys $(K_{\text{enc}}, K_{\text{prf}}, K_1, K_2, \dots, K_{\lfloor n/2 \rfloor})$ using standard key-separation techniques. Each key $K_i = (K_i[1], \dots, K_i[r])$, where $r \geq 3$ is a constant and each $K_i[t] \in \mathcal{K}$. Now when $|M| \geq n$, proceed as we have described above, using keys K_{enc} and K_{prf} to encipher M . But when $|M| < n$ then encipher M using an r -round Feistel network. If $|M| = 2i$ then for the t -th “round function” f_t of the Feistel construction use $f(x) = F_{K_i[t]}(x \parallel 0^i)[1..i]$. Using the results of [8] (or subsequent, analytically stronger results) this construction can be proven sound.

Enciphering messages of odd bit length would seem to have no practical importance (messages are almost invariably byte strings). But if it is desired to encipher such strings one can always use a slightly unbalanced Feistel network and the same approach described above.

3.4 Analysis

The following theorem says that F as constructed above is a secure variable-input-length cipher, as long as both G and S are secure.

Theorem 3.2 Let G be a parsimonious PRF, let S a parsimonious encryption scheme, and let $F = F[G, S]$ be the resulting variable-input-length cipher as described above. Then

$$\text{Sec}_F^{\text{prp}}(t, q, \mu) \leq \text{Sec}_G^{\text{prf}}(t', q, \mu) + \text{Sec}_S^{\text{priv}}(t', q, \mu) + \frac{q^2}{2^n},$$

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

PROOF OF THEOREM. Let A be an adversary attacking F , and let t be its running time, q the number of queries it makes, and μ the total length of all its queries put together. We assume that A never repeats an oracle query. (This is wlog, since, if not, construct A' which runs A and stores the oracle queries made by A and the answers received. Whenever A makes a query, A' first checks if it was already made, and if so returns the right answer from its stored list. Only if the query is new does A' actually go make it of the oracle. Note that this works because the oracle supplied to A is deterministic.) This is important to some of the claims made below, which will not hold if A repeats a query. We consider various probabilities related to running A under various different experiments:

$$\begin{aligned} p_1 &= \Pr[K \leftarrow \mathcal{K} : A^{F_K(\cdot)} = 1] \\ p_2 &= \Pr[K_{\text{enc}} \leftarrow \mathcal{K}_{\text{enc}} ; g \leftarrow \text{Rand}(\mathcal{M}, n) : A^{\text{Encrypt}_{K_{\text{enc}}}(\cdot)_{\text{prefix}}; g(M)} = 1] \\ p_3 &= \Pr[K_{\text{enc}} \leftarrow \mathcal{K}_{\text{enc}} ; A^{\text{Encrypt}_{K_{\text{enc}}}(\cdot)_{\text{prefix}}} = 1] \end{aligned}$$

$$\begin{aligned}
p_4 &= \Pr[K_{\text{enc}} \leftarrow \mathcal{K}_{\text{enc}} ; A^{\text{Encrypt}_{K_{\text{enc}}}(\mathbb{S}^{(\cdot)}_{\text{prefix}})} = 1] \\
p_5 &= \Pr[\pi \leftarrow \text{Perm}(\mathcal{M}) : A^{\pi(\cdot)} = 1].
\end{aligned}$$

Our goal is to upper bound $\text{Adv}_A^{\text{PRP}}(F) = p_1 - p_5$. We do this in steps.

Claim 3.3 $p_1 - p_2 \leq \text{Sec}_G^{\text{prf}}(t', q, \mu)$.

Proof: Consider the following distinguisher D for G . It has an oracle for $g: \mathcal{M} \rightarrow \{0, 1\}^n$. It picks $K_{\text{enc}} \leftarrow \mathcal{K}_{\text{enc}}$. It runs A , and when A makes oracle query M it returns $\text{Encrypt}_{K_{\text{enc}}}(M_{\text{prefix}}; g(M))$ to A as the answer. Finally D outputs whatever A outputs. Then

$$\begin{aligned}
\Pr[K_{\text{prf}} \leftarrow \mathcal{K}_{\text{prf}} : D^{G_{K_{\text{prf}}(\cdot)}} = 1] &= p_1 \\
\Pr[g \leftarrow \text{Rand}(\mathcal{M}, n) : D^{g(\cdot)} = 1] &= p_2.
\end{aligned}$$

So $\text{Adv}_D^{\text{prf}}(G) = p_1 - p_2$. The claim follows. \blacksquare

Claim 3.4 $p_2 = p_3$

Proof: The only difference between the experiment underlying p_2 and that underlying p_3 is that in the former, the IV used for encryption is a random function of M , while in the latter it is chosen at random by the encryption algorithm. These are the same as long as all the oracle queries are different, which is what we assumed about A . \blacksquare

Claim 3.5 $p_3 - p_4 \leq \text{Sec}_S^{\text{priv}}(t', q, \mu)$.

Proof: Consider the following adversary B for S that is given an oracle \mathcal{O} . It runs A , and when A makes oracle query M it returns $\mathcal{O}(M_{\text{prefix}})$ to A as the answer. Finally D outputs whatever A outputs. Then

$$\begin{aligned}
\Pr[K_{\text{enc}} \leftarrow \mathcal{K}_{\text{enc}} : B^{\text{Encrypt}_{K_{\text{enc}}}(\cdot)} = 1] &= p_3 \\
\Pr[K_{\text{enc}} \leftarrow \mathcal{K}_{\text{enc}} : B^{\text{Encrypt}_{K_{\text{enc}}}(\mathbb{S}^{(\cdot)})} = 1] &= p_4.
\end{aligned}$$

So $\text{Adv}_B^{\text{priv}}(S) = p_3 - p_4$. The claim follows. \blacksquare

Claim 3.6 $p_4 - p_5 \leq q^2/2^n$.

Proof: Let

$$r = \Pr[h \leftarrow \text{Rand}(\mathcal{M}) : A^{h(\cdot)} = 1].$$

We argue that $p_4 - r \leq q^2/2^{n+1}$ and also $r - p_5 \leq q^2/2^{n+1}$. The claim follows by the triangle inequality. It remains to prove the two subclaims.

The second subclaim, that $r - p_5 \leq q^2/2^{n+1}$, is of course clear; the statistical distance between a family of functions and a family of permutations is given by the collision probability under q queries. So consider the first subclaim, namely $p_4 - r \leq q^2/2^{n+1}$. This is true because the encryption scheme is parsimonious. The IV is chosen at random, and for each fixed IV, the map $\text{Encrypt}_{K_{\text{enc}}}((\cdot)_{\text{prefix}}; \text{IV})$ is a permutation on \mathcal{M} . Thus, $p_4 - r$ is the statistical distance between a family of permutations on \mathcal{M} and a family of random functions on \mathcal{M} , which is again $q^2/2^{n+1}$ because all strings in \mathcal{M} have length at least n . \blacksquare

Given these claims, we can complete the proof of the theorem by noting that

$$\text{Adv}_A^{\text{PFP}}(F) = p_1 - p_5 = (p_1 - p_2) + (p_2 - p_3) + (p_3 - p_4) + (p_4 - p_5) .$$

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.. Much of Phil's work on this paper was carried out while on sabbatical at Chiang Mai University, Thailand, hosted by the Computer Service Center, under Prof. Krisorn Jittorntrum and Prof. Darunee Smawatakul.

References

- [1] R. ANDERSON AND E. BIHAM, "Two Practical and Provably Secure Block Ciphers: BEAR and LION." *Fast Software Encryption 3*, 1996, Springer-Verlag LNCS 1039.
- [2] 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.
- [3] M. BELLARE, J. KILIAN AND P. ROGAWAY, "On the security of cipher block chaining." *Advances in Cryptology – Crypto 94 Proceedings*, Lecture Notes in Computer Science Vol. 839, Y. Desmedt ed., Springer-Verlag, 1994.
- [4] M. BLAZE, "High-bandwidth encryption with low-bandwidth smartcards." *Fast Software Encryption '96*. Springer Verlag (1996).
- [5] O. GOLDBREICH, S. GOLDWASSER AND S. MICALI, "How to construct random functions." *Journal of the ACM*, Vol. 33, No. 4, 210–217, (1986).
- [6] S. GOLDWASSER AND S. MICALI, "Probabilistic encryption." *Journal of Computer and System Sciences* **28**, 270-299, April 1984.
- [7] ISO/IEC 9797, "Information technology – Security techniques – Data integrity mechanism using a cryptographic check function employing a block cipher algorithm," International Organization for Standardization, Geneva, Switzerland (1994) (second edition).
- [8] M. LUBY AND C. RACKOFF, "How to construct pseudorandom permutations from pseudorandom functions." *SIAM J. Computing*, Vol. 17, No. 2, April 1988.
- [9] U. MAURER, "A simplified and generalized treatment of Luby-Rackoff pseudorandom permutation generators." *Advances in Cryptology – Eurocrypt 92 Proceedings*, Lecture Notes in Computer Science Vol. 658, R. Rueppel ed., Springer-Verlag, 1992, pp. 239–255.
- [10] S. MICALI, C. RACKOFF AND R. SLOAN, "The notion of security for probabilistic cryptosystems." *SIAM J. Computing*, Vol. 17, No. 2, April 1988.

- [11] M. NAOR AND O. REINGOLD, “On the construction of pseudorandom permutations: Luby-Rackoff revisited.” *Proceedings of the 29th Annual Symposium on Theory of Computing*, ACM, 1997.
- [12] National Bureau of Standards, FIPS PUB 46, “Data Encryption Standard.” U.S. Department of Commerce, January 1977.
- [13] National Bureau of Standards, FIPS PUB 81, “DES Modes of Operation.” U.S. Department of Commerce, December 1980.
- [14] J. PATARIN, “Improved security bounds for pseudorandom permutations.” Manuscript (1997).
- [15] E. PETRANK AND C. RACKOFF, “CBC MAC for Real-Time Data Sources,” manuscript, available at <http://philby.ucsd.edu/cryptolib.html>, 1997.
- [16] J. PIEPRZYK, “How to construct pseudorandom permutations from single pseudorandom functions.” *Advances in Cryptology – Eurocrypt 90 Proceedings*, Lecture Notes in Computer Science Vol. 473, I. Damgård ed., Springer-Verlag, 1990 pp. 140–150.
- [17] B. PRENEEL AND P. VAN OORSCHOT, “MDx-MAC and building fast MACs from hash functions,” *Advances in Cryptology – Crypto 95 Proceedings*, Lecture Notes in Computer Science Vol. 963, D. Coppersmith ed., Springer-Verlag, 1995, 1–14.
- [18] R. RIVEST, “All-or-nothing encryption and the package transform.” *Fast Software Encryption '97*, Springer-Verlag (1997).
- [19] C. SHANNON, “Communication theory of secrecy systems.” *Bell Systems Technical Journal*, 28(4), 656–715 (1949).
- [20] Y. ZHENG, T. MATSUMOTO AND H. IMAI, “Impossibility results and optimality results on constructing pseudorandom permutations.” *Advances in Cryptology – Eurocrypt 89 Proceedings*, Lecture Notes in Computer Science Vol. 434, J-J. Quisquater, J. Vandewille ed., Springer-Verlag, 1989.