# All-Or-Nothing Encryption and The Package Transform


Ronald L. Rivest

MIT Laboratory for Computer Science
545 Technology Square, Cambridge, Mass. 02139
`rivest@theory.lcs.mit.edu`



**Abstract.** We present a new mode of encryption for block ciphers, which we call *all-or-nothing encryption*. This mode has the interesting defining property that one must decrypt the *entire* ciphertext before one can determine *even one* message block. This means that brute-force searches against all-or-nothing encryption are slowed down by a factor equal to the number of blocks in the ciphertext. We give a specific way of implementing all-or-nothing encryption using a "package transform" as a pre-processing step to an ordinary encryption mode. A package transform followed by ordinary codebook encryption also has the interesting property that it is very efficiently implemented in parallel. All-or-nothing encryption can also provide protection against chosen-plaintext and related-message attacks.


## 1 Introduction

One way in which a cryptosystem may be attacked is by brute-force search: an adversary tries decrypting an intercepted ciphertext with all possible keys until the plaintext "makes sense" or until it matches a known target plaintext. Our primary motivation is to devise means to make brute-force search more difficult, by appropriately pre-processing a message before encrypting it.

In this paper, we assume that the cipher under discussion is a block cipher with fixed-length input/output blocks, although our remarks generalize to other kinds of ciphers. An "encryption mode" is used to extend the encryption function to arbitrary length messages (see, for example, Schneier [9] and Biham [3]).

In general, the work required to search for an unknown $k$-bit key to a known block cipher is $2^k$ in the worst-case, or $2^{k-1}$ on the average. Here (and throughout this paper) we measure the work by the number of elementary decryptions attempted, where an elementary decryption is a decryption of one block of ciphertext. For example, in the "electronic codebook" encryption mode the adversary needs to decrypt only the first block of ciphertext to obtain the first block of plaintext; this is usually sufficient to identify the correct key. (If not, the second block can be decrypted as well...)

Sometimes the size of the key space for one's encryption algorithm is fixed, "marginal," and can't be improved. For example, one can argue that a 56-bit DES key is marginal (see Blaze et al. [4]). Or, one may be encumbered by export regulations that restrict one to a 40-bit secret key. The question posed here is: *is there any way to significantly increase the difficulty for an adversary of performing a brute-force search, while keeping the key size the same and not overly burdening the legitimate communicants?*

We show that the answer to the question is *yes*.

## 2 Strongly non-separable encryption

The problem with most popular encryption modes is that the adversary can obtain one block of plaintext by decrypting *just one* block of ciphertext.

We illustrate this point with cipher-block chaining (CBC mode). Let the $s$ blocks of the message be denoted $m_1$, $m_2$, ..., $m_s$. The CBC mode utilizes an initialization vector $IV$ and a key $K$. The algorithm produces as output ciphertext $c_i$ for $1 \leq i \leq s+1$, where

$$c_1 = IV$$

and

$$c_{i+1} = E(K, c_i \oplus m_i) \text{ for } i = 1, 2, \ldots, s \ .$$

Thus

$$m_i = c_i \oplus D(K, c_{i+1}) \text{ for } i = 1, 2, \ldots, s \ ,$$

and so any one of the $s$ message blocks can be obtained with the decryption of just one ciphertext block. This makes the adversary's key-search problem relatively easy, since decrypting a single ciphertext block is generally enough to test a candidate key.

Let us say that an encryption mode for a block cipher is *separable* if it has the property that an adversary can determine one block of plaintext by decrypting just one block of ciphertext. Thus, CBC mode is separable.

We wish to design non-separable encryption modes. More precisely, we wish to design *strongly* non-separable modes, defined as follows.

**Definition.** Suppose that a block cipher encryption mode transforms a sequence

$$m_1, m_2, \ldots, m_s$$

of $s$ message blocks into a sequence

$$c_1, c_2, \ldots, c_t$$

of $t$ ciphertext blocks, for some $t$, $t \geq s$. We say that the encryption mode is *strongly non-separable* if it is infeasible to determine even one message block $m_i$ (or any property of a particular message block $m_i$) without decrypting *all* $t$ ciphertext blocks.

## 3 All-Or-Nothing Transforms

We propose to achieve strongly non-separable modes as follows:

- Transform the message sequence $m_1, m_2, \ldots, m_s$ into a "pseudo-message" sequence $m'_1, m'_2, \ldots, m'_{s'}$ (for some $s' \geq s$) with an "all-or-nothing transform", and
- Encrypt the pseudo-message with an ordinary encryption mode (e.g. codebook mode) with the given cryptographic key $K$ to obtain the ciphertext sequence $c_1, c_2, \ldots, c_t$.

We call encryption modes of this type "all-or-nothing encryption modes." A specific instance of this mode would be "all-or-nothing codebook mode," when the encryption mode used is codebook mode, (or "all-or-nothing CBC mode", etc.).

To make this work, the all-or-nothing transform has to have certain properties.

**Definition.** A transformation $f$ mapping a message sequence $m_1, m_2, \ldots, m_s$ into a pseudo-message sequence $m'_1, m'_2, \ldots, m'_{s'}$ is said to be an *all-or-nothing transform* if

- The transformation $f$ is reversible: given the pseudo-message sequence, one can obtain the original message sequence.
- Both the transformation $f$ and its inverse are efficiently computable (that is, computable in polynomial time).
- It is computationally infeasible to compute any function of any message block if any one of the pseudo-message blocks is unknown.

We note that an all-or-nothing transformation must really be randomized, so that a chosen or known message attack does not yield a known pseudo-message, and so that a deterministic function which computes the first pseudo-message block is not available as a function to contradict the last requirement above.

We note that the all-or-nothing transformation is not itself "encryption," since it makes no use of any secret key information. It is merely an invertible "preprocessing" step that has certain interesting properties. The actual encryption in an all-or-nothing encryption mode is the operation that encrypts the pseudo-message resulting from the all-or-nothing transform. An all-or-nothing transform is a fixed public transform that anyone can perform on the message to obtain the pseudo-message, or invert given the pseudo-message to obtain the message.

**Theorem 1.** *An all-or-nothing encryption mode is strongly non-separable.*

**"Proof":** We assume that the underlying encryption mode is such that all ciphertext blocks must be decrypted in order to obtain all pseudo-message blocks. (If this were not the case, the encryption mode would not be efficient, and a more efficient reduced mode could be derived from it.) Thus, all ciphertext blocks must be decrypted in order to determine any (property of any) message block. □

## 4 The Package Transform

The all-or-nothing scheme we propose here (the "package transform") is quite efficient, particularly when the message is long; the cost of an all-or-nothing transform is approximately twice the cost of the actual encryption. We shall also see that all-or-nothing encryption admits fast parallel implementations.

The legitimate communicants thus pay a penalty of approximately a factor of three in the time it takes them to encrypt or decrypt in all-or-nothing mode, compared to an ordinary separable encryption mode. However, an adversary attempting a brute-force attack pays a penalty of a factor of $t$, where $t$ is the number of blocks in the ciphertext.

As an example, if I send you a eight-megabyte message encrypted in all-or-nothing CBC mode with a 40-bit DES key, the adversary must decrypt the entire eight-megabyte file in order to test a single candidate 40-bit key. This expands the work-factor by a factor of one-million, compared to breaking ordinary CBC mode. Since one million is approximately $2^{20}$, to the adversary this feels like having to break a 60-bit key instead of a 40-bit key!

Using this scheme, it can clearly be advantageous for the communicants to "pad" the message with random data, as it makes the adversary's job harder.

We propose here a particular all-or-nothing transform, which we call the "package transform." We note that while it uses a block cipher itself as a primitive, no secret keys are used. (Instead, a randomly chosen key is used, and this key can be easily determined from the pseudo-message sequence.) The block cipher used in the package transform need not be the same as the block cipher used to encipher the pseudo-message (the package transform output), although it may be. (If it is the same encryption algorithm, note that we assume below that the key space for the package transform block cipher is sufficiently large that brute-force search is infeasible, while the motivation for the use of an all-or-nothing encryption mode was that the key space for the outer encryption algorithm was marginal. This situation can arise for variable-key-length block ciphers such as RC5. For concreteness, the reader may imagine that we are working with RC5 for both the package transform encryption algorithm and the outer encryption algorithm, with 128-bit input/output blocks, a 128-bit encryption key for the package transform, and a 40-bit key for the outer encryption transform.)

For this exposition, then, we assume that the key size of the package transform block cipher is the same as its block size; this assumption can easily be removed and is made here only for convenience in exposition. We also assume that the key space for the package transform block cipher is sufficiently large that brute-force searching for a key is infeasible. The scheme also uses a fixed publically-known key $K_0$ for the package transform block cipher.

Here is the package transform:

- Let the input message be $m_1, m_2, \ldots, m_s$.
- Choose at random a key $K'$ for the package transform block cipher.
- Compute the output sequence $m'_1, m'_2, \ldots, m'_{s'}$ for $s' = s + 1$ as follows:
  - Let $m'_i = m_i \oplus E(K', i)$ for $i = 1, 2, 3, \ldots, s$.

- Let
$$m'_{s'} = K' \oplus h_1 \oplus h_2 \oplus \cdots \oplus h_s \ ,$$
where
$$h_i = E(K_0, m'_i \oplus i) \text{ for } i = 1, 2, \ldots, s \ ,$$
where $K_0$ is a fixed, publically-known encryption key.

The intent here is that the key $K'$ be chosen from a large space (for example, chose $K'$ as a 128-bit RC5 key). Since $K'$ is not a secret shared key (it is disclosed in the pseudo-message), it is not restricted by the limitations of the following encryption mode.

The package transformation is similar to encrypting in counter mode, except that the key is randomly chosen rather than fixed, and the last pseudo-message block is the exclusive-or of the key and a hash of all previous pseudo-message blocks (computed as the exclusive-or of the encryptions of variants of these blocks under a fixed key, where the $i$-th variant is computed as the exclusive-or of $i$ and the block). This technique ensures that simple modifications to the ciphertext, such as permuting the order of two blocks or duplicating a blocks, is highly likely to change the key $K'$ computed by the receiver.

One could also define variant package transforms based on block-chaining techniques instead of counter mode.

It is easy to see that the package transform is invertible:
$$K' = m'_{s'} \oplus h_1 \oplus h_2 \oplus \cdots \oplus h_s \ ,$$
$$m_i = m'_i \oplus E(K', i) \text{ for } i = 1, 2, \ldots, s \ .$$

We also note that if any block of the pseudo-message sequence is unknown, then $K'$ can not be computed, and so it is infeasible to compute any message block. (Formal proof omitted here, but we recall that the key $K'$ is assumed to be drawn from an infeasibly large set, so that (for example) a meet-in-the-middle attack is not more efficient than decrypting all the ciphertext blocks.)

## 5 Discussion

A related well-known approach towards getting more security out of fixed number of key bits is to use encryption techniques that have a long "set-up" time (see Quisquater et al. [8], or Schneier's "Blowfish" algorithm [9]). This penalizes the legitimate user whenever he performs a key-change, whereas all-or-nothing encryption incurs a fixed penalty for each block encrypted. While this may seem to favor the increased set-up time approach, we note that

- An all-or-nothing transform is merely a pre-processing step, and so it can be used with already-existing encryption devices and software, without changing the encryption algorithm.

– Increasing the set-up time may still yield an algorithm that is efficiently implemented with a special-purpose brute-force chip, since there may be little need for inter-chip communications. On the other hand, the two-pass nature of all-or-nothing encryption may necessitate large amounts of input/output, something that usually slows down operations considerably.
– In any case, the approaches are complementary, and can easily be combined.

We note that all-or-nothing encryption modes are only defined here when the message to be encrypted is a finite sequence; an infinitely long message can not be encrypted in an all-or-nothing mode, whereas other modes such as CBC work perfectly well in this case. All-or-nothing encryption modes work very well in cases such as for encrypting packets in a network.

We observe, however, that one can begin encrypting in package CBC mode (or package codebook mode) before one knows the end of message sequence, since the inner package operation and the outer CBC (or codebook) encryption modes can both be implemented in a sequential manner. However, decrypting a package mode ciphertext more-or-less requires two passes and/or having the entire ciphertext available at once.

Package codebook mode is particularly interesting, since the outer codebook decryption and the inner package transformation can both be performed efficiently in parallel. (I don't mean that they are performed at the same time, but that each one separately admits an efficient parallel implementation.) With a sufficient number of encryption units, a message of length $s$ can be encrypted or decrypted in time $O(\log s)$. This may be an advantage for the legitimate communicants in a high-speed communications scenario. Note that the same advantage is available to the adversary–although he has to decrypt the entire ciphertext, he can also do it in parallel. However, for the adversary this advantage is probably meaningless, since it is the total search time that is important to him, not the latency for performing a single decryption. Thus package codebook mode has much to recommend it from a performance perspective.

We note that all-or-nothing encryption modes can provide protection against differential attacks and other forms of attack that depend on chosen plaintext, since a randomized all-or-nothing transformation can effectively destroy any patterns in the actual input (the pseudo-message) to the underlying encryption operation.

In addition, an all-or-nothing transformation can be useful before RSA encryption, as it prevents various kinds of "related message" or other attacks (e.g. those of Coppersmith et al. [5]). Indeed, the package transform described here can be viewed as a special case of the "simple embedding scheme" proposed by Bellare and Rogaway [2] in their "optimal asymmetric encryption" preprocesing scheme (used before applying RSA encryption):

$$x \oplus G(r) \mid\mid r \oplus H(x \oplus G(r)) \ .$$

Here $x$ is the message to be encrypted (like our message $m$), $r$ is a randomly chosen quantity (like our key $K'$), $G(r)$ is a pseudo-random output (like our $E(K',1)$, $E(K',2)$, $\ldots$), and $H$ is a hash function (like our $h_1 \oplus h_2 \oplus \ldots h_s$).

The correspondence would be closer if we had proposed using $m'_{s'} = K' \oplus MD5(m'_1, \ldots, m'_s)$, which would also give some improved efficiency, but we wished to confine ourselves to just using the block cipher as a primitive operation. We are applying these ideas to symmetric block cipher modes of operation rather than asymmetric encryption, but the principles are essentially the same. However, it may also be the case that a rather different approach can be applied to achieve our goals with substantially greater efficiency than the approach suggested here or by Bellare and Rogaway's approach in general.

There are many approaches one might take towards devising all-or-nothing transforms. One might consider computing the pseudomessage as the concatenation of a description of a hash function $h$ chosen randomly from a universal family of hash functions with a suitably large range, followed by the application of $h$ to the message. Another approach that may work well is to use a scheme based on an FFT-like arrangement of randomized multipermutations (see Schnorr et al. [10]).

Or, one can base an approach on secret-sharing schemes. Actually, the package transform can be viewed as a $s'$ out of $s'$ secret-sharing threshold scheme; each of the $s'$ pseudo-message blocks can be viewed as one "share" of the underlying message. Decrypting so as to obtain fewer than $s'$ pseudo-message blocks yields no information at all about the underlying message. This is "computational secret sharing" (see [6]) since the shares are shorter than the message itself. Indeed, one can design all-or-nothing schemes based on Krawczyk's proposals.

An entirely different approach is given by Anderson and Biham [1], who design block ciphers (such as BEAR and LION) from scratch that seem to have an "all-or-nothing" property. Their approach is different because they design block ciphers with variable-length blocks to accomodate messages of varying lengths, whereas our focus is on designing an encryption mode for fixed-length block ciphers that provide an all-or-nothing property. Nonetheless, their schemes may be the method of choice in some situations.

We note that all-or-nothing encryption has *terrible* error-propagation properties: if *any* ciphertext block is damaged, then it is likely that *every* message block will be damaged. Thus, ciphertext should be transported with reliable transmission means. (One could interpose an error-correction phase between the all-or-nothing transformation and the encryption; this could help handle errors while only modestly decreasing non-separability.)

Using this error-propagation property to one's advantage, one can extend all-or-nothing mode by appending a suitable block of redundancy (such a block of all zeros, or the sum of all the previous message blocks) to the message before applying the all-or-nothing transformation. This redundancy can be verified and removed upon decryption. This helps to detect corrupted ciphertext.

As a variation on the idea of the previous paragraph, the redundancy block may be computed as the sum of previous message blocks and a secret value that is known only to the two parties communicating; this provides a form of message authentication. The redundancy block could of course also be computed with more conventional keyed hashing techniques.

The preceding paragraphs touch upon an important issue: that an encryption mode should provide *integrity* as well as *confidentiality*. Mao and Boyd [7] make this point well. Bellare and Rogaway prove that their simple embedding scheme provides non-malleability, for example.

## 6 Conclusion

We have presented an encryption mode—the all-or-nothing encryption mode—and a specific means of implementing it using the package transform. Other forms of all-or-nothing encryption are presumably yet to be devised.

We leave it as an open problem to devise an all-or-nothing encryption mode that is substantially more efficient than the scheme presented here. Is it possible, for example, to reduce the cost of implementing an all-or-nothing mode from a factor of three greater than CBC to just a factor of two greater?

**Acknowledgments**

I would like to thank Don Coppersmith, Oded Goldreich, Shafi Goldwasser, Mihir Bellare, Burt Kaliski, and the referees for helpful comments and conversations. Silvio Micali deserves special thanks for suggesting the term "all-or-nothing." David Wagner deserves thanks for pointing out significant bugs in earlier versions of this paper, and for pointing out the relationship between this work and the Bellare-Rogaway work on optimal asymmetric encryption. And thanks to Mihir Bellare for noting the relationship with secret-sharing schemes.

## References

1. Ross Anderson and Eli Biham. Two practical and probably secure block ciphers: BEAR and LION. In Dieter Gollman, editor, *Fast Software Encryption*, pages 114–120. Springer, 1996. (Proceedings Third International Workshop, Feb. 1996, Cambridge, UK).
2. Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption—how to encrypt with RSA. In *EUROCRYPT94*, 1994.
3. Eli Biham. Cryptanalysis of multiple modes of operation. 1995. Pre-Proceedings of ASIACRYPT '94. Submitted to J. Cryptology.
4. Matt Blaze, Whitfield Diffie, Ronald L. Rivest, Bruce Schneier, Tsutomu Shimomura, Eric Thompson, and Michael Wiener. Minimal key lengths for symmetric ciphers to provide adequate commercial security: A report by an ad hoc group of cryptographers and computer scientists, January 1996. Available at `http://www.bsa.org`.
5. Don Coppersmith, Matthew Franklin, Jacques Patarin, and Michael Reiter. Low-exponent RSA with related messages. Technical Report IBM RC 20318, IBM T.J. Watson Research Lab, December 27, 1995. (To appear in Eurocrypt '96).
6. Hugo Krawczyk. Secret sharing made short. In Douglas R. Stinson, editor, *Proc. CRYPTO 93*, pages 136–146. Spring-Verlag, 1993.
7. Wenbo Mao and Colin Boyd. Classification of cryptographic techniques in authentication protocols. In *Proceedings 1994 Workshop on Selected Areas in Cryptography*, May 1994. (Kingston, Ontario, Canada).

8. J.-J. Quisquater, Yvo Desmedt, and Marc Davio. The importance of “good” key scheduling schemes (how to make a secure DES scheme with $\leq 48$ bit keys). In H. C. Williams, editor, *Proc. CRYPTO 85*, pages 537–542. Springer, 1986. Lecture Notes in Computer Science No. 218.
9. Bruce Schneier. *Applied Cryptography (Second Edition)*. John Wiley & Sons, 1996.
10. C. P. Schnorr and S. Vaudenay. Black box cryptanalysis of hash networks based on multipermutations. In *EUROCRYPT94*, 1994.

This article was processed using the LaTeX macro package with LLNCS style