

CONTRIBUTION TO: IEEE P1363

TITLE: Encryption of Long Blocks Using a Short-Block Encryption Procedure

SOURCE: Don B. Johnson^Φ, Certicom
Stephen M. Matyas, IBM
Mohammad Peyravian^Ψ, IBM

DATE: November 1996

ABSTRACT:

This contribution presents a formatting method for encrypting a plaintext block using a block encryption algorithm (such as Elliptic Curve, RSA, DES, etc.) having a block size smaller than that of the plaintext block. The process of encrypting a plaintext block consists of first masking the plaintext block to form a masked plaintext block and then encrypting a portion or all of the masked plaintext block. The masking method is a reversible procedure which performs a "complete" mixing of the plaintext block such that no bit in the plaintext block can be determined unless every bit in the masked plaintext block is known. We recommend that the scheme presented in this contribution be adopted into the baseline text as a formatting scheme for encrypting a long plaintext block using a short-block encryption algorithm.

NOTICE:

This contribution has been prepared to assist the IEEE P1363 standard body. This proposal is made by the authors as a basis of discussion. This contribution should not be construed as a binding proposal on the authors or their companies. Specifically, the authors and their companies reserve the right to amend or modify the statements contained herein.

^Φ This work was done while Don Johnson was working at IBM.

^Ψ Contact person - Mohammad Peyravian

Email: peyravn@vnet.ibm.com
Phone: +1-919-254-7576

1. Introduction

The IEEE P1363 committee in its August 1996 meeting decided to seek recommendations on formatting methods for encrypting a long plaintext block using a block encryption algorithm having a block size smaller than that of the plaintext block. This contribution presents one such formatting method.

The proposed scheme consists of two steps: first, a masking operation is performed on the input plaintext to produce a masked plaintext. The masking method is a reversible procedure which performs a “complete” mixing of the plaintext block such that no bit in the plaintext block can be determined unless every bit in the masked plaintext block is known. Then, a portion or all of the masked plaintext is encrypted using a block encryption algorithm (such as Elliptic Curve, RSA, DES, etc.). Since the scheme requires only a portion of the block to be encrypted, the scheme has lower computational complexity compared to the common asymmetric-key encryption schemes (such as Elliptic Curve in which the input block has to be divided into sub-blocks of the key length first and then each sub-block has to be EC encrypted separately). As long as the plaintext contains sufficient secret data and the encryption key is long enough, the proposed scheme provides the same level of protection as if the entire plaintext was encrypted.

The formatting method described in this contribution is similar to a degree to the method of masking described by Bellare-Rogaway in Optimal Asymmetric Encryption [1] and by Johnson-Le-Martin-Matyas-Wilkins in the IBM method implemented in the Transaction Security System [2, 3]. However, the JLMMW method calls for one iteration of masking, the BR method calls for two iterations of masking, whereas the method presented in this contribution calls for at least four iterations of masking. Less than four iterations of the masking process are needed if the secret portion of the input plaintext block and the encrypted portion of the masked plaintext block are structured in certain ways. But, to make the masking process a general “black box” that would apply to all cases with no requirement on the structure of the secret value of the plaintext block and the encrypted portion of the masked plaintext block, four iterations of the masking process are needed. The masking process prevents an adversary from being able to invert the unencrypted part of the masked plaintext block to determine part or all of the input plaintext block. The masking process makes each bit in the masked plaintext block a function of each and every bit in the input plaintext block.

2. Long Plaintext Block Encryption

The encryption of a plaintext block (PlaintextBlock) of length m bits consists of two steps: 1) a masking process, and 2) an encryption process. As illustrated in Figure 1, first the PlaintextBlock (X) goes through a reversible masking function which produces a Masked PlaintextBlock (Y) of the same length as the PlaintextBlock. Then, an n -bit portion of the Masked PlaintextBlock, where $n \leq m$, is encrypted using an encryption algorithm (such as Elliptic Curve, RSA, DES, etc.).

2.1 Masking Process

The masking process consists of dividing the PlaintextBlock (X) into a left-hand part L and a right-hand part R. Parts L and R are of equal or near equal length. A balanced construct is preferable from a security viewpoint as it provides optimal distribution of the PlaintextBlock

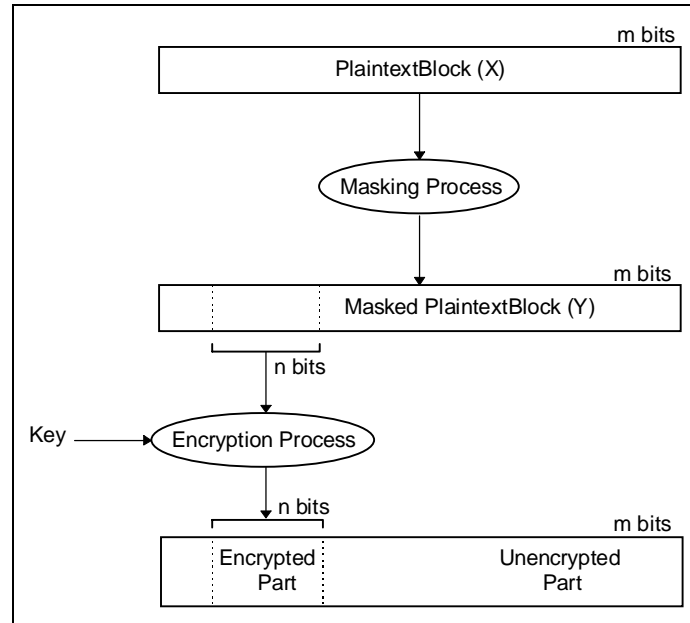


Figure 1: Long plaintext block encryption

random secret data throughout the Masked PlaintextBlock. The process of masking consists of the following steps:

1. Masking L with R to produce mL.
2. Masking R with mL to produce mR.
3. Masking mL with mR to produce mmL.
4. Masking mR with mmL to produce mmR.

Additional iterations of masking can be performed if desired. Figure 2 illustrates the masking process.

In the first iteration, a mask value $H(R)$ is calculated on R using a strong collision resistant one-way hash function H. The length of $H(R)$ is equal to the length of L. Then, $H(R)$ is Exclusive-ORed with L to produce mL.

In the second iteration, a mask value $H(mL)$ is calculated on mL using the hash function H. The length of $H(mL)$ is equal to the length of R. Then, $H(mL)$ is Exclusive-ORed with R to produce mR.

In the third iteration, a mask value $H(mR)$ is calculated on mR using the hash function H. The length of $H(mR)$ is equal to the length of mL. Then, $H(mR)$ is Exclusive-ORed with mL to produce mmL.

In the fourth iteration, a mask value $H(mmL)$ is calculated on mmL using the hash function H. The length of $H(mmL)$ is equal to the length of mR. Then, $H(mmL)$ is Exclusive-ORed with mR to produce mmR.

mmL is the final mask of L and mmR is the final mask of R. The concatenation of mmL and mmR forms the Masked PlaintextBlock (Y).

The unmasking process is simply the inverse of the masking process. In this case, the iterations are performed in the reverse order (i.e., 4, 3, 2, and 1). At iteration 4, $H(mmL)$ is calculated and is Exclusive-ORed with mmR to produce mR. At iteration 3, $H(mR)$ is calculated and is

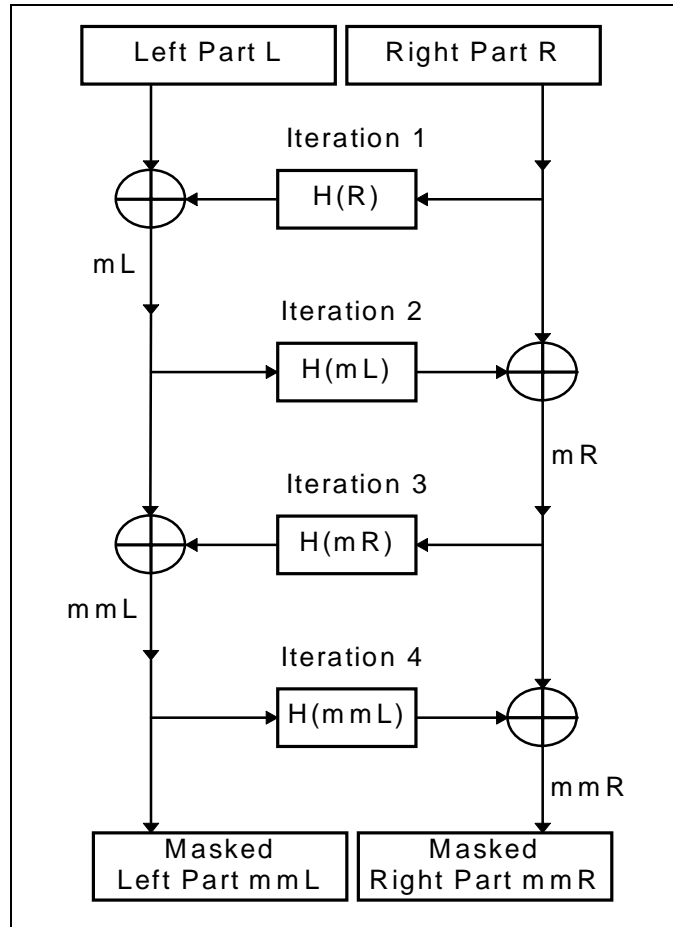


Figure 2: Masking process

Exclusive-ORed with mmL to produce mL . At iteration 2, $H(mL)$ is calculated and is Exclusive-ORed with mR to produce the original right-hand input R . At iteration 1, $H(R)$ is calculated and is Exclusive-ORed with mL to produce the original left-hand input L .

2.1.1 Number of Iterations of Masking Process

When H is a strong collision resistant one-way hash function, four iterations of the masking process is sufficient to make each bit in the Masked PlaintextBlock (Y) dependent on each and every bit in the PlaintextBlock (X). This achieves complete intersymbol dependence. However, if H is not a strong collision resistant one-way hash function, additional iterations of the masking process may need to be performed. For example, SHA-1 is believed to be a strong collision resistant one-way hash function. So, with SHA-1 as the hash function, four iterations of the masking process appears to be sufficient.

Less than four iterations of the masking process are needed if the random secret value S of X and the hidden (i.e., encrypted) portion of Y are structured in certain ways. But, to make the masking process a general “black box” that would apply to all cases with no requirement on the structure of the random secret value S of X and the hidden (i.e., encrypted) portion of Y , four iterations of the masking process are needed. For example, with three iterations of the masking process one can recover the random secret value S from the public parts of X and Y , if $n \leq s \leq \|L\|/2$ (where $\|L\|$ is the length of L in bits, n is the number of encrypted, bits in the

Masked PlaintextBlock, and s is the number of bits in the random secret value S , S is in one side of L , and the encrypted bits are in mL but in the opposite side of S .

2.1.2 Implementation of the Masking Process

The function H is assumed to be a strong collision resistant one-way hash function. One possible implementation of H would be to employ a strong collision resistant one-way hash function such as SHA-1 that calculates a k -bit (where $k = 160$ bits for SHA-1) output value from an i -bit input value.

Suppose, for example, that we use a one-way hash function such SHA-1 for H . Then the masking process can be implemented as follows. First the m -bit PlaintextBlock is divided into a left part L and a right part R . The length of L and R are defined as follows: 1) if m is even, then L and R each contain $m/2$ bits, and 2) if m is odd, then L contains $(m-1)/2$ bits and R contains $(m+1)/2$ bits.

Figure 3 illustrates the masking process for iteration one with a one-way hash function. L is first divided into SubBlocks of k bits. If the length of L is a multiple of k bits, then each SubBlock contains k bits. If the length of L is not a multiple of k bits, then L will have one short SubBlock of less than k bits. The division of L into sub-blocks will produce r SubBlocks, where $r \geq 1$. Next, for each SubBlock in L , a separate hash value is calculated on R . The hash value is then Exclusive-ORed with the corresponding SubBlock of L to produce a Masked SubBlock of L . The masking operation consists of the following steps:

1. A counter value $1, 2, \dots, r$ (matching the SubBlock number in L to be masked) is concatenated with R . Some informational values (represented by "etc" in Figure 3) may also be concatenated with R . A hash is then computed on the concatenated values and is Exclusive-ORed with the corresponding SubBlock of L to produce a Masked SubBlock of L of length k

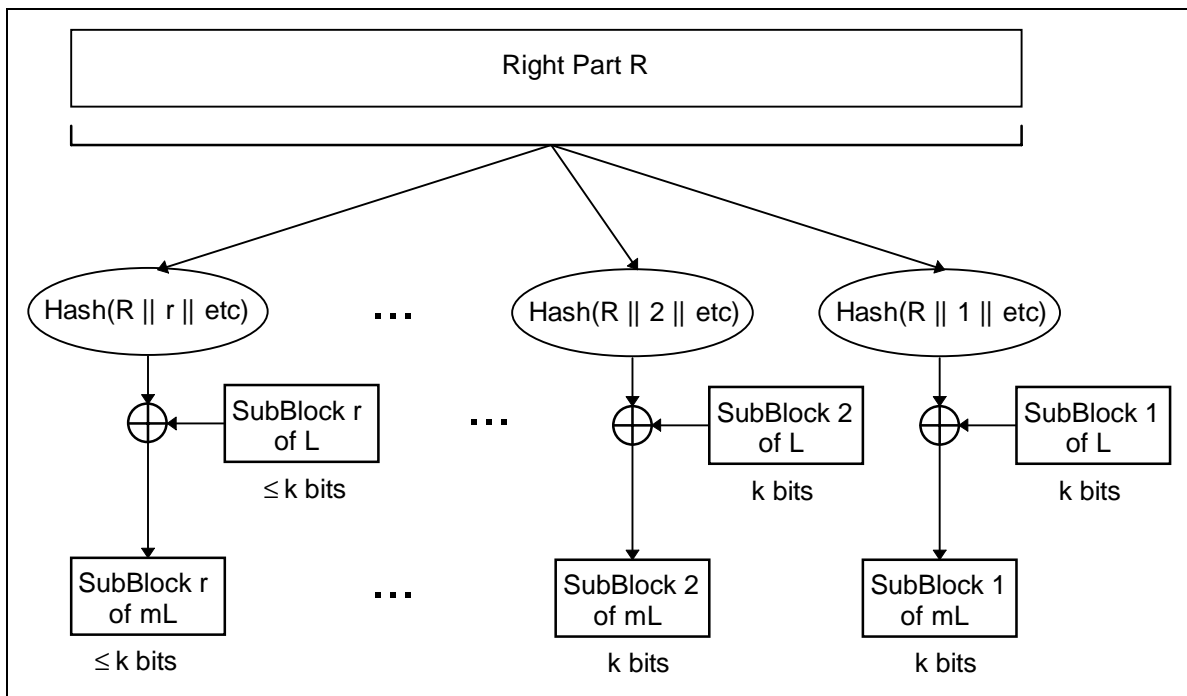


Figure 3: Iteration one of masking process with a hash function of length k bits

- bits. If the last SubBlock r of L is a short SubBlock containing j bits ($j < k$), then SubBlock r is masked by Exclusive-ORing it with the least significant j bits of the hash value.
2. The r Masked SubBlocks of L are concatenated to produce mL .

The masking operation for iterations two, three, four, etc. are performed the same way. Note that the masking process preserves the block length, i.e., the output block from the masking process has the same length as the input block.

2.2 Encryption Process

An n -bit portion of the Masked PlaintextBlock, where $n \leq m$, is encrypted using an encryption algorithm. The remaining $m-n$ bits of the Masked PlaintextBlock is left unencrypted. The encrypted part can be any portion of the Masked PlaintextBlock.

The encryption can be done using any form of block encryption algorithm (such as Elliptic Curve, RSA, DES, etc.) having an n -bit block size (e.g., using a 160-bit key Elliptic Curve encryption). The encryption function may consist of a simple Exclusive-OR function, i.e., n bits of the Masked PlaintextBlock is encrypted by Exclusive-ORing it with an n -bit shared secret key.

3. Evaluation of the Proposed Scheme

In this section, we discuss the advantages of the scheme and the design issues that need to be considered.

3.1 Advantages

The presented scheme has the following advantages:

1. The reversible masking scheme is built on the “feistel-ladder” (with a strong collision resistant one-way hash function) achieves complete intersymbol dependence with the following cryptographic properties:
 - ⇒ Each bit in the Masked PlaintextBlock is a function of each and every bit in the PlaintextBlock.
 - ⇒ No bit in the PlaintextBlock can be determined unless every bit in the Masked PlaintextBlock is known.
 - ⇒ The Masked PlaintextBlock can be protected by encrypting any portion of it, as long as the PlaintextBlock contains sufficient secret data and the encryption key is long enough.
 - ⇒ The scheme is encryption algorithm independent, i.e., it can be used with any asymmetric- and symmetric-key block encryption algorithm (e.g., Elliptic Curve, RSA, DES, etc.).
 - ⇒ The masking scheme works for any size block and preserves the block length, i.e., the output block from the masking process has the same length as the input block. This would have the possible advantage that developing standards would not have to pay attention to block sizes that may differ from one algorithm to another.
2. Since the scheme requires only a portion of the block to be encrypted, the scheme has lower computational complexity compared to the common asymmetric-key encryption schemes

(such as Elliptic Curve in which the input block has to be divided into sub-blocks of the key length first and then each sub-block has to be EC encrypted separately).

3.2 Design Issues

The presented scheme requires that the following design issues be considered.

1. The presented scheme requires that the input plaintext contain sufficient secret data to prevent exhaustive attacks from being carried out to find its value. Suppose the input PlaintextBlock is X and the Masked PlaintextBlock is Y , each of length m bits. Assume that X contains a secret value S of length s bits (where $s \leq m$) and that we encrypt n bits of Y (where $n \leq m$). Given the non-secret part of X and the unencrypted part of Y , an adversary may try three types of attack:
 - ⇒ **Exhaust on S:** If s is small, the adversary can pick a trial value of S , which gives a trial value of X . He masks X and compares the output with the $m-n$ bits in the unencrypted part of Y . If he finds a match, he has a trial value of S that can be checked using other pairs [of] (non-secret part of X , unencrypted part of Y).
 - ⇒ **Exhaust on encrypted part of Y:** If n is small, the adversary makes a trial guess for the masked plaintext in Y corresponding to the encrypted part of Y . He then un.masks Y and compares the recovered value of X with the non-secret part of X . If he finds a match, he has a trial value of S that can be checked using other pairs [of] (non-secret part of X , unencrypted part of Y).
2. As stated in No. 1 above, the input PlaintextBlock contains sufficient secret data to prevent exhaustive attacks from being carried out. The present contribution does not specify or define the means by which the secret information is put or placed in the input PlaintextBlock. There are several ways in which this could be accomplished. However, the masking and encryption process described herein does not depend on how this is accomplished, and therefore is not limited to any particular mechanism or method for accomplishing same. The Optimal Asymmetric Encryption (OAE) technique (Bellare and Rogaway) calls for a secret random number to be appended to the input, which is one possible way.

In this contribution we presented a formatting procedure for encrypting a long plaintext block using a short-block encryption algorithm. If the group decides to accept this scheme, we volunteer to submit a contribution with all the necessary details for inclusion in the baseline text.

References

- [1] M. Bellare and P. Rogaway, "Optimal Asymmetric Encryption - How to Encrypt with RSA," Eurocrypt '94 Proceedings, volume 950 of Lecture Notes in Computer Science, Springer-Verlag, New York.
- [2] D. Johnson, A. Le, W. Martin, S. Matyas, and J. Wilkins, "Hybrid Key Distribution Scheme Giving Key Record Recovery," IBM Technical Disclosure Bulletin, 37(2A), 5-16, February 1994.
- [3] D. Johnson and S. Matyas, "Asymmetric Encryption: Evolution and Enhancements," CryptoBytes, volume 2, number 1, Spring 1996.