Encryption Modes with Almost Free Message Integrity

Charanjit S. Jutla
IBM T.J. Watson Research Center

In this document we propose two new modes of operation for symmetric key block cipher algorithms. The main feature distinguishing the two proposed modes from existing modes is that along with providing confidentiality of the message, they also provide message integrity. In other words, the new modes are not just modes of operation for encryption, but modes of operation for *authenticated encryption.* As the title of the document suggests, the new modes achieve the additional property with little extra overhead, as will be explained below.

One of the new modes is highly parallelizable. In fact, the parallelizable mode has critical paths of only two block cipher invocations. By one estimate, a hardware implementation of this mode on a single board (housing 1000 block cipher units) achieves terabits/sec (10^{12} bits/sec) of authenticated encryption. Moreover, there is no penalty for doing a serial implementation of this mode.

The new modes also come with **proofs of security**, assuming that the underlying block ciphers are secure. For confidentiality, the modes achieve the same provable security bounds as CBC. For authentication, the modes achieve the same provable security bounds as CBC-MAC.

The non-parallelizable mode is similar to the CBC mode. It differs from the CBC mode in that the output is “whitened” (XORed) with a pairwise independent random sequence. The pairwise independent sequence can be generated with little overhead as detailed below. It is this whitening with the pairwise independent sequence that assures message integrity.

The parallelizable mode removes the chaining from the above mode, and instead does an input whitening (in addition to the output whitening) with a pairwise independent sequence. Thus, it becomes similar to the ECB mode. However, with the input whitening with the pairwise independent sequence the new mode has provable security similar to CBC (Note: ECB does not have security guarantees like CBC).

Both the parallelizable mode and the non-parallelizable mode come in two flavors. These flavors refer to how the pairwise independent sequence is generated. In one mode, the pairwise independent sequence is generated by a subset construction. In another mode, the pairwise independent sequence is generated by (ai+b) modulo a fixed prime number. There will be one standard prime number for each bit-size block cipher. Thus, for 64 bit block ciphers the prime could be $2^{64}-257$. For 128 bit block ciphers, the prime could be $2^{128}-159$.

The modes are described below in more detail. For proofs of security see http://eprint.iacr.org/2000/039.ps .

We first describe the non-parallelizable mode.

1. Integrity Aware Cipher Block Chaining Mode (IACBC)

Let n be the block size of the underlying block cipher. If the block cipher requires keys of length k, then this mode requires two independent keys of length k. Let these keys be called K0 and K1. From now on, we will use f_K to denote the encryption function under key K.

The message to be encrypted P, is divided into blocks of length n each. Let these blocks be $P_1, P_2, \ldots P_{m-1}$. As in CBC, a random initial vector r of length n bits is chosen. This random vector is expanded into $t = \lceil \log(m+1) \rceil$ new random vectors using the block cipher and key K0 as follows:

```
for i= 0 to t-1 do
    $W_i = f_{K0}(r+i+1)$
end for
```

The t random and independent vector are used to prepare m ($\leq 2^t-1$) new pairwise independent random vectors $S_0, S_1, \ldots, S_{m-1}$. This can be done by taking all subsets of $W_1, W_2, \ldots, W_t$, and for each subset taking their xor-sum. There is a well-known fast way to generate these subsets, and xor-sums (also known as Gray Code: http://hissa.nist.gov/dads/HTML/graycode.html). The following pseudocode is the *proposed sequence* of generating these subsets. This code also incorporates the generation of W_i above.

```
$procedure\ \ pairwise\_independent\_sequence(in\ \ r, m, K0;\ \ out\ S)\{$
$S_{-1} = 0;$
$for\ i = 0\ to\ m-1\ do$
    $j = i+1;$
    $k = 0;$
    $/*\ find\ the\ index\ of\ the\ least\ significant\ ON\ bit\ in\ (i+1)\ */$
    $while\ ((j\ \&1) == 0)\ do$
        $k = k+1;\ j = j >> 1;\ \ /*increment\ k\ and\ right\ shift\ j\ */$
    $end\ while$
    $if\ ((j \oplus 1) == 0)\ \ \ \ \ \ /*if\ (i+1)\ is\ a\ power\ of\ 2\ */$
        $W_k = f_{K0}(r+k+1);$
    $S_i = S_{i-1} \oplus W_k;$
$end\ for\}$
```

Note that, S_i is obtained from S_{i-1} with just one XOR. The inner while loop condition is checked two times on average.

The ciphertext message C=<C0,C1,…,Cm> is generated as follows (see fig 1):

```
$M_0 = r$
$N_0 = f_{K1}(M_0)$
$C_0 = N_0$
$for\ \ i = 1\ to\ m-1\ do$
    $M_i = P_i \oplus N_{i-1}$
    $N_i = f_{K1}(M_i)$
    $C_i = N_i \oplus S_i$
$end\ \ for$
$checksum = \sum_{i=1}^{m-1} P_i$
$M_m = checksum \oplus N_{m-1}$
$N_m = f_{K1}(M_m)$
$C_m = N_m \oplus S_0$
```

Again, the summation is an xor-sum. Note that S_0 is used in the last step.
The above scheme is invertible. The inversion process yields blocks $P_1, P_2, \ldots, P_m$. The decrypted plaintext is $<P_1, P_2, \ldots P_{m-1}>$. Message integrity is ***verified*** by checking $P_m = \sum_{i=1}^{m-1} P_i$.

```
$Decryption \quad pseudocode:$

$N_0 = C_0$
$M_0 = f^{-1}{}_{K1}(N_0)$
$r = M_0$
$invoke \quad pairwise\_independent\_sequence(r, m, K0, S);$
$for\ i = 1\ to\ m-1\ do$
    $N_i = C_i \oplus S_i$
    $M_i = f^{-1}{}_{K1}(N_i)$
    $P_i = M_i \oplus N_{i-1}$
$end\ \ for$
$checksum = \sum_{i=1}^{m-1} P_i$
$N_m = C_m \oplus S_0$
$M_m = f_{K1}(M_m)$
$P_m = M_m \oplus N_{m-1}$
$Integrity \equiv (P_m == checksum)$
```

Alternative Method for Generating Pairwise Independent Sequence

The pairwise independent sequence can be generated by a simple algebraic method as well. This could be an ***alternate*** standard.
Let p=2^{128}-159. The number p is known to be a prime. For a 128-bit block cipher, instead of generating log m new random numbers, just two new random numbers are generated.

```
$a = f_{K0}(r+1)$
$b = f_{K0}(r+2)$
$if\ \ b > (2^{128} - 159) \quad then \quad b = (b+159) \bmod 2^{128}$
```

Having generated a,b, the sequence $S_0, S_{1,\ldots} S_{m-1}$ is generated as follows:

```
$S_0 = a$
$for\ i = 1\ to\ m-1\ do$
    $S_i = (S_{i-1} + b) \bmod 2^{128}$
    $if\ \ (b > S_i) \quad S_i = S_i + 159$
$end\ \ for$
```

For 64-bit ciphers p=2^{64}-257 is recommended.

2. Integrity Aware Parallelizable Mode (IAPM)

The pairwise independent sequence is generated as in the non-parallelizable mode IACBC, by invoking pairwise_independent_sequence(r,m+1,K0,S). The ciphertext C=<C0,C1,…,Cm> however is generated differently (see fig 2) as follows:

```
$M_0 = r$
$N_0 = f_{K1}(M_0)$
$C_0 = N_0$
$for\ i = 1\ to\ m-1\ do$
    $M_i = P_i \oplus S_i$
    $N_i = f_{K1}(M_i)$
    $C_i = N_i \oplus S_i$
$end\ \ for$
$checksum = \sum_{i=1}^{m-1} P_i$
$M_m = checksum \oplus S_m$
$N_m = f_{K1}(M_m)$
$C_m = N_m \oplus S_0$
```

Note that the same pairwise independent sequence is used for both the input whitening and the output whitening.

Again, the decryption process is straightforward. The inversion process yields blocks $P_1,P_2,\ldots,P_m$. The decrypted plaintext is $<P_1,P_2,\ldots P_{m-1}>$. Message integrity is ***verified*** by checking

$$P_m = \sum_{i=1}^{m-1} P_i \ .$$

3. Performance

The IACBC scheme was implemented for DES on IBM PowerPC (200MHz). For messages of size 1024 64-bit blocks the IACBC scheme(with the subset construction) yielded throughput of 31.5 Mbits/sec. In comparison, the CBC scheme (i.e. just encryption) ran at 33.78 Mbits/sec. We expect similar performance results for serial implementations of IAPM.

4. Patents

IBM has filed a U.S. patent on all these schemes.

Figure 1 (IACBC)

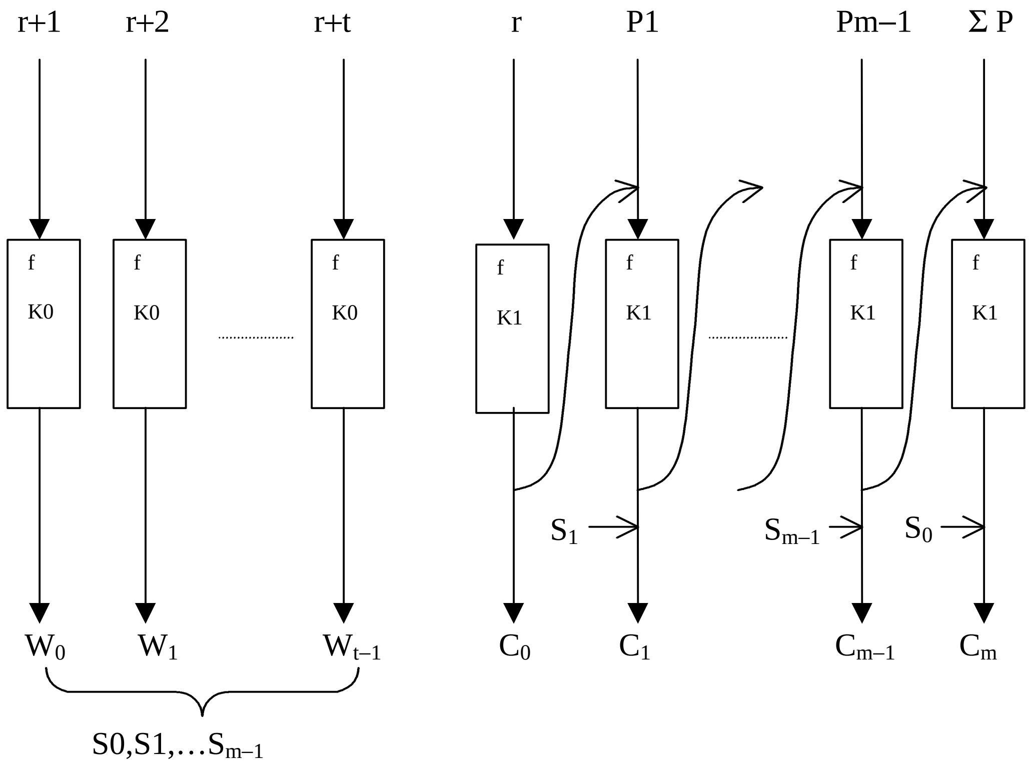
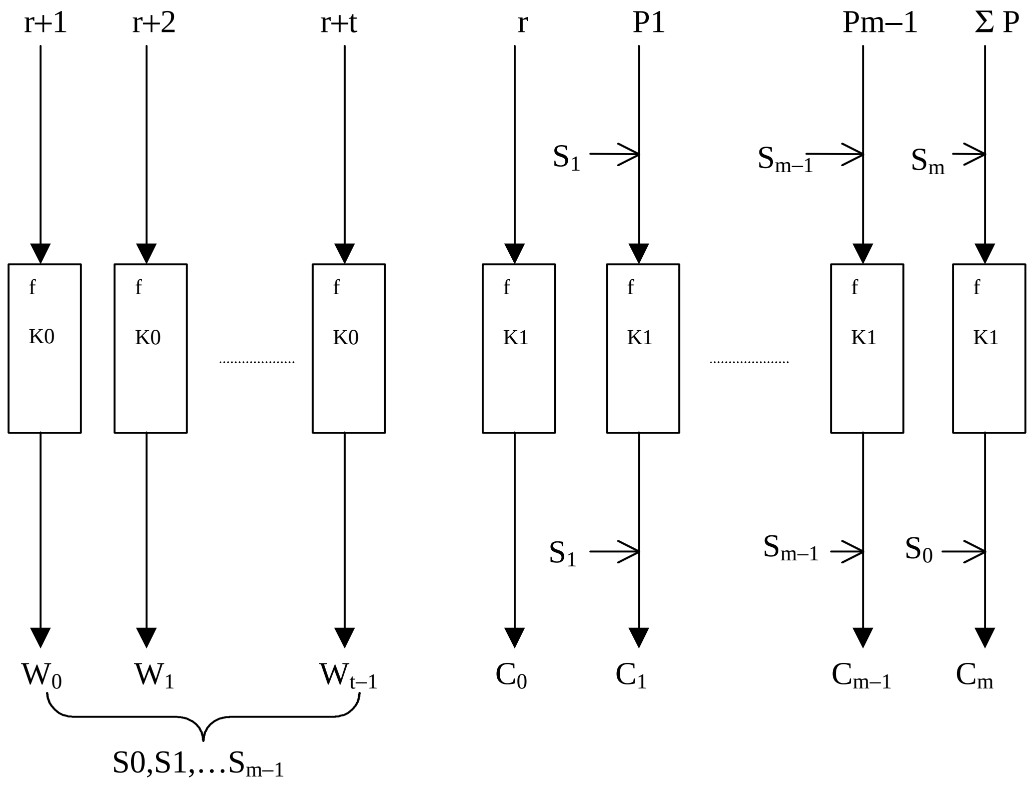


Figure 2 (IAPM)


r+1
r+2
r+t
r
P1
Pm−1
Σ P
S1
Sm−1
Sm
f
K0
f
K0
f
K0
f
K1
f
K1
f
K1
f
K1
S1
Sm−1
S0
W0
W1
Wt−1
C0
C1
Cm−1
Cm
S0,S1,…Sm−1