Analysing the IOBC Authenticated Encryption Mode

Chris J. Mitchell

Information Security Group, Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK
`c.mitchell@rhul.ac.uk`

Abstract. The idea of combining a very simple form of added plaintext redundancy with a special mode of data encryption to provide data integrity is an old one; however, despite its wide deployment in protocols such as Kerberos, it has largely been superseded by provably secure authenticated encryption techniques. In this paper we cryptanalyse a block cipher mode of operation called IOBC, possibly the only remaining encryption mode designed for such use that has not previously been analyzed. We show that IOBC is subject to known-plaintext-based forgery attacks with a complexity of around $2^{n/3}$, where n is the block cipher block length.

1 Introduction

This is perhaps the last chapter in a long and rather unfortunate story[1], namely that of 'special' modes of operation for block ciphers, designed to offer 'low cost' combined integrity and confidentiality protection by combining encryption with the addition of simple (or fixed) redundancy to the plaintext. The underlying idea is to design the mode so that modifying the ciphertext without invalidating the added redundancy is impossible without knowledge of the encryption key. It is a long story since the idea dates back over 30 years, and a sad story because one by one these special modes of operation have been shown to fail to meet their objectives.

Such modes are the theme of section 9.6.5 of Menezes, van Oorschot and Vanstone's landmark book [1]. As they point out, the starting point for the development of such modes is the observation that encryption alone does not guarantee integrity. This, combined with the observation that the 'obvious' approach of encrypting the data and then separately computing a MAC involves twice the work, leads to the alternative notion of adding detectable redundancy before encrypting so that it can be detected after decryption. Two main methods for adding redundancy have been proposed:

- add a fixed block at the end of the plaintext, which may be public or secret (in the latter case the block acts as an auxiliary key);

[1] Since ACISP has played its own part in this developing tale, it seems fitting to present this final chapter at ACISP 2013.

– append to the plaintext some easily computed and simple (public) function of the plaintext.

In either case we refer to the block added to the end of the plaintext as a *Manipulation Detection Code (MDC)*. Whichever approach is adopted, the method for computing the MDC needs to be simple, or it offers no advantage over the more conventional 'encrypt then MAC' approach.

Note that there is a third, related, approach which remains viable and is, indeed, increasingly used; this involves computing a keyed function of the plaintext (a sort of 'weak MAC') which is then encrypted to make it secure. The plaintext itself may or may not be encrypted. Indeed, one example of such an approach, namely GCM/GMAC [2], has a proof of security and has been standardized [3, 4]. The main differences between GCM (and other related techniques) and the approaches we are concerned with here is that GCM uses a 'standard' encryption method and, of course, GCM has a proof of security.

At this point we observe that the general approaches described above possess an intrinsic weakness if the method of adding redundancy is public[2]. Suppose an attacker persuades the legitimate originator of protected messages to encrypt a message containing a correct MDC somewhere in the middle (where the MDC is correct in the sense that it is a correct MDC on the data that precedes it). The attacker will then be able to delete all ciphertext blocks following the encrypted MDC, and such a change will not be detectable. Despite this weakness, using an appropriate encryption mode combined with a public method for adding verifiable redundancy to a message has been widely used for message integrity protection — e.g. in Kerberos (see, for example, [5]). We thus restrict our attention in the remainder of the paper to the case where the MDC is a secret value (this is the approach proposed for use with the IOBC mode, which forms the main subject of this paper).

Regardless of the nature of the MDC, the method of encryption needs to be chosen with great care. Using a stream cipher or CBC mode encryption is clearly totally insecure, since a ciphertext produced with such a technique can readily be manipulated in such a way that large parts of the recovered plaintext, including the end of the plaintext, remains unchanged. A simple modified version of CBC called Plaintext Cipher Block Chaining (PCBC) [6, 7], in which the feedback chaining variable is the exclusive-or of the immediately preceding ciphertext and plaintext blocks, was proposed back in the 1980s to avoid this problem. This scheme was used in Kerberos version 4 [7] to provide combined encryption and integrity protection. The weakness of PCBC for use as an integrity-protection mode was first pointed out by Kohl [6]. As is simple to verify, Kohl showed that interchanging two of the ciphertext blocks of a PCBC-encrypted message does not affect the decryption of the final block, i.e. it is extremely simple to make undetectable changes to messages. Note that this is actually a stronger attack than is implied by Menezes, van Oorschot and Vanstone [1] who refer only to the danger of known-plaintext attacks.

[2] This problem appears to be part of the cryptographic folklore — it was pointed out to the author by Bart Preneel in the late 1990s

Menezes, van Oorschot and Vanstone [1] describe a slightly different mode, confusingly also referred to as PCBC (this time for *Plaintext-Ciphertext Block Chaining*), in which the feedback chaining variable is the modulo 2^n sum of the immediately preceding ciphertext and plaintext blocks. This technique was also described as long ago as 1982 by Meyer and Matyas [8]. Unfortunately, despite its long history, the latter version of PCBC was shown to possess major weaknesses when used with an MDC for integrity protection in a 2005 ACISP paper [9]. M-PCBC, a further variant of PCBC due to Sierra et al. [10], was also shown to fail to offer appropriate MDC protection in the same ACISP 2005 paper [9]. Another variant of PCBC was proposed by Gligor and Donescu [11]; however, this latter scheme, known as iaPCBC, was shown to possess serious vulnerabilities by Ferguson et al. [12]. Yet another scheme, called PES-PCBC, was proposed by Zuquete and Guedes [13] but, as observed by Recacha [14] as well the authors themselves [15], PES-PCBC is subject to known-plaintext attacks.

Indeed, to the author's knowledge, almost all the proposals for such special modes have been cryptanalyzed, with one exception — a scheme proposed in 1996 called *Input and Output Block Chaining (IOBC)* [14]. One possible reason why IOBC has escaped attention is that until recently the only available description was in Spanish. However, recently an English language translation of the original 1996 paper[3] has been made available by the author, and it is this translation that has been used as the basis of this paper. It is interesting to note that IOBC has, nevertheless, had an impact on the cryptographic literature. A modified version of IOBC, called EPBC, was published in 1997 [15] and was subsequently cryptanalyzed in 2007 [16].

Finally we observe that, although the history of the area sketched above may give the impression of an ordered and coherent narrative, the truth is somewhat different. Ideas, and sometimes attacks, appear to have been put forward independently of one another, and one role of this introduction is to try to pull together the main developments in the area.

The remainder of the paper is structured as follows. Section 2 provides details of the operation of IOBC and its use for integrity protection. This is followed in section 3 by certain observations on the properties of IOBC and its component functions. This sets the stage for section 4, in which a known plaintext forgery attack against IOBC when used for integrity protection is described. A brief discussion of possible fixes to IOBC is given in section 5, before a certificational chosen-plaintext attack on IOBC is outlined in section 6. Concluding remarks are provided in section 7.

[3] http://inputoutputblockchaining.blogspot.com.es/

2 The Recacha IOBC Mode

We next describe the operation of the IOBC block cipher mode of operation[4]. We base our description on Recacha's 1996 paper [14], although we use the notation of Mitchell [16].

2.1 Initial Assumptions

We suppose that IOBC is to be used with an n-bit block cipher, i.e. a block cipher operating on plaintext and ciphertext blocks of n bits. We further suppose that n is a power of two, and put $n = 2^m$ (as is the case for all standardized block ciphers [17]). We write $e_K(P)$ for the result of block cipher encrypting the n-bit block P using the secret key K, and $d_K(C)$ for the result of block cipher decrypting the n-bit block C using the key K. Finally we suppose the plaintext to be protected is divided into a sequence of n-bit blocks (if necessary, having first been padded): $P_1, P_2, \ldots, P_t$.

2.2 Initialization Vectors

The scheme uses two secret n-bit Initialization Vectors (IVs), denoted by F_0 and G_0. The nature of the intended restrictions on their use is not altogether clear. However, one suggestion in the original Recacha paper [14] is that they should be generated as follows.

Suppose K' is an auxiliary key used solely for generating the IVs. Suppose also that S is a sequence number, managed so that different values are used for every message. Then $F_0 = e_{K'}(S)$ and $G_0 = e_{K'}(F_0)$. For the purposes of this paper we assume that F_0 and G_0 are always generated this way, and thus the scheme can be thought of as employing a pair of block cipher keys and a non-secret, non-repeating, sequence number (which must be carefully managed to prevent accidental re-use of sequence number values). Note that special measures will need to be taken if the same key is to be used to encrypt communications in both directions between a pair of parties. Avoiding sequence number re-use in such a case could be achieved by requiring one party to start the sequence number they use for encryption at a large value, perhaps halfway through the range.

2.3 Operation

The IOBC encryption of the plaintext $P_1, P_2, \ldots, P_t$ operates as follows:

$$\begin{aligned}
G_i &= P_i \oplus F_{i-1}, \quad (1 \le i \le t),\\
F_i &= e_K(G_i), \quad (1 \le i \le t),\\
C_i &= F_i \oplus g(G_{i-1}), \quad (2 \le i \le t),
\end{aligned}$$

[4] PES-PCBC (briefly mentioned in section 1) is the same as IOBC with the exception that in PES-PCBC the function g is the identity function.

where $C_1 = F_1 \oplus G_0$, $\oplus$ denotes bit-wise exclusive-or, and g is a function that maps an n-bit block to an n-bit block, defined below. The operation of the mode (when used for encryption) is shown in Figure 1. Note that we refer to the values F_i and G_i as 'internal' values, as they are computed during encryption, but they do not constitute part of the ciphertext.

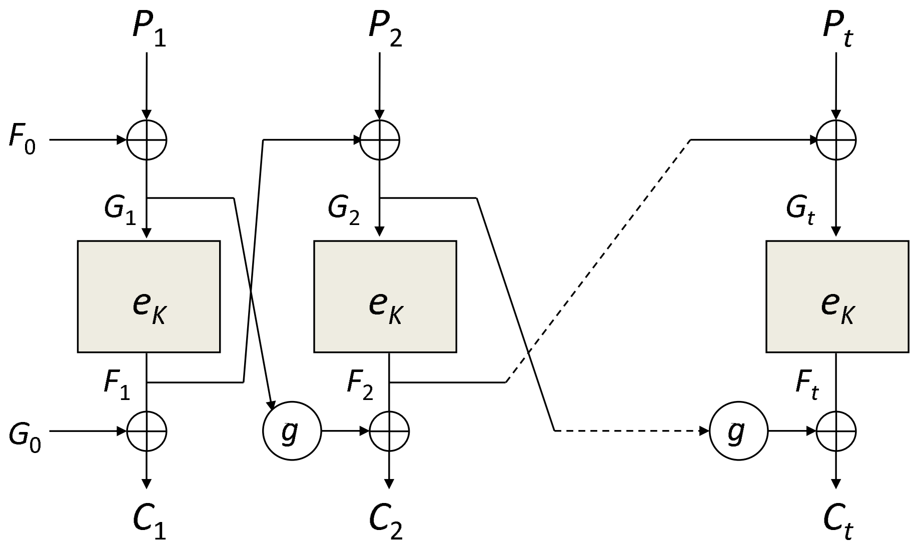

Fig. 1. IOBC encryption

The function g is defined as follows. Suppose X is an n-bit block, where $n = 2^m$. Suppose also that $X = L||R$ where L is the leftmost $2^{m-1} - 1$ bits of X and R is the rightmost $2^{m-1} + 1$ bits of X (and, as throughout, $||$ denotes concatenation). Then

$$g(X) = (>_1 (L))||(>_1 (R))$$

where $>_i$ denotes a rightwards (cyclic) shift by i bit positions.

Decryption operates similarly. We have:

$$\begin{aligned} F_i &= C_i \oplus g(G_{i-1}), \quad (2 \le i \le t), \\ G_i &= d_K(F_i), \quad (1 \le i \le t), \\ P_i &= G_i \oplus F_{i-1}, \quad (1 \le i \le t). \end{aligned}$$

and $F_1 = C_1 \oplus G_0$, where d denotes block cipher decryption.

2.4 Additional Remarks

As described above, we assume throughout that the IVs F_0 and G_0 are derived by ECB-mode-encrypting a sequence number using a secondary key. Thus the ciphertext blocks will be a function of this serial number (as well as the pair of keys used). We thus write $[S], C_1, C_2, \ldots, C_t$ for a sequence of ciphertext blocks, meaning that $C_1, C_2, \ldots, C_t$ were encrypted using the sequence number S. This is logical, since the sequence number will need to be sent or stored with the ciphertext to enable correct decryption.

Finally observe that IOBC should only be used with relatively short messages. That is, as specified by Recacha [14] (and for reasons which become clear below), a message to be encrypted using IOBC shall contain at most $n^2/2 - 1$ plaintext blocks, where n is the plaintext block length. Thus for $n = 64$ and $n = 128$, the two most commonly used block lengths, a message shall contain at most 2047 and 8191 blocks, respectively.

2.5 Using IOBC for Integrity Protection

As already implied, IOBC is designed for combined confidentiality and integrity protection. Confidentiality comes simply by encrypting the data using IOBC mode. Integrity is achieved by adding an MDC to the end of the plaintext — what Recacha [14] refers to as an *Integrity Check Vector (ICV)*. After decryption of an IOBC-protected message, the receiver must check that the ICV is correct, and must reject the message if it is not.

Recacha recommends use of a secret ICV of length $n/2$. This ICV must clearly be known to the intended recipient, and should therefore be regarded as forming part of the key (along with the key K used in IOBC computations and the key K' used to derive the IVs).

3 Preliminary Observations

We first establish some simple results on the operation of the IOBC scheme. The first Lemma derives directly from a discussion in section 6 of [16]. It is also implicit in the discussions of Recacha [14].

Lemma 1. *Suppose $[S], C_1, C_2, \ldots, C_t$ and $[S'], C'_1, C'_2, \ldots, C'_{t'}$ are IOBC encrypted versions of the plaintext sequences $P_1, P_2, \ldots, P_t$ and $P'_1, P'_2, \ldots, P'_{t'}$, respectively. If the ciphertext:*

$$\begin{gathered}[S'], C^*_1, C^*_2, \ldots, C^*_{t-v+u} = \\ [S'], C'_1, C'_2, \ldots, C'_{u-1}, C_v \oplus g(G'_{u-1}) \oplus g(G_{v-1}), C_{v+1}, \ldots, C_t\end{gathered}$$

is submitted for IOBC decryption (where $1 < u$ and $1 < v < t$, and G_{v-1} and G'_{u-1} are values computed during the encryption of the respective sequences of blocks), then the resulting sequence of plaintext blocks $P^*_1, P^*_2, \ldots, P^*_{t-v+u}$ will be equal to

$$P'_1, P'_2, \ldots, P'_{u-1}, P_v \oplus F'_{u-1} \oplus F_{v-1}, P_{v+1}, P_{v+2}, \ldots, P_t.$$

Proof We first note that it follows immediately from the definitions that $F_i^* = F_i'$ and $G_i^* = G_i'$ $(1 \leq i \leq u-1)$, where F_i' and G_i' are the internal values generated during the encryption process that yielded the ciphertext message $C_1', C_2', \ldots, C_{t'}'$. Hence $P_i^* = P_i'$ $(1 \leq i \leq u-1)$.

We now consider the decryption of C_u^*. We have:

$$\begin{aligned} F_u^* &= C_u^* \oplus g(G_{u-1}^*) \quad \text{(from section 2.3)} \\ &= C_v \oplus g(G_{u-1}') \oplus g(G_{v-1}) \oplus g(G_{u-1}^*) \quad \text{(by defn. of } C_u^*) \\ &= C_v \oplus g(G_{v-1}) \quad \text{(since } G_{u-1}^* = G_{u-1}') \\ &= F_v \quad \text{(from section 2.3).} \end{aligned}$$

Hence $G_u^* = G_v$. Finally we have:

$$\begin{aligned} P_u^* &= G_u^* \oplus F_{u-1}^* \quad \text{(from section 2.3)} \\ &= G_v \oplus F_{u-1}' \quad \text{(from above)} \\ &= P_v \oplus F_{u-1}' \oplus F_{v-1} \quad \text{(from section 2.3).} \end{aligned}$$

We now consider the decryption of C_{u+1}^*. We have

$$\begin{aligned} F_{u+1}^* &= C_{u+1}^* \oplus g(G_u^*) \quad \text{(from section 2.3)} \\ &= C_{v+1} \oplus g(G_v) \quad \text{(by defn. of } C_{u+1}^* \text{ and from above)} \\ &= F_{v+1} \quad \text{(from section 2.3).} \end{aligned}$$

Hence $G_{u+1}^* = G_{v+1}$. Finally, we have

$$\begin{aligned} P_{u+1}^* &= G_{u+1}^* \oplus F_u^* \quad \text{(from section 2.3)} \\ &= G_{v+1} \oplus F_v \quad \text{(from above)} \\ &= P_{v+1} \quad \text{(from section 2.3).} \end{aligned}$$

The same argument shows that $P_{u+i}^* = P_{v+i}$ for every $i \geq 1$, and the desired result follows. □

Remark 1. Lemma 1 suggests a way in which it may be possible to forge an IOBC-encrypted message so that the final block will contain the correct ICV. However, the problem remains of discovering $g(G_{u-1}') \oplus g(G_{v-1})$ (as used in constructing the message in the statement of the lemma). Recacha [14] discusses this very point, and explains that making this difficult motivates the inclusion of the function g in the design of IOBC — that is, if g was not included (as is the case for PES-PCBC), then simple forgeries could be achieved by manipulating a single encrypted message for which part of the plaintext was known. We revisit this point later, and show that g is not as effective in achieving the goal as intended.

We next give some elementary observations on the operation of IOBC.

Lemma 2. *Suppose* $[S], C_1, C_2, \ldots, C_t$ *is the encryption of* $P_1, P_2, \ldots, P_t$ *using IOBC, and that* F_i *and* G_i *are as defined in section 2.3. Then:*

(i) $C_{j+1} \oplus P_{j+2} = g(G_j) + G_{j+2}, \;\; 1 \leq j \leq t-2;$
(ii) $\bigoplus_{i=1}^{k} g^{k-i}(C_{j+2i-1} \oplus P_{j+2i}) = g^k(G_j) \oplus G_{j+2k}, \;\; 1 \leq j \leq t-2, \; 1 \leq k \leq (t-j)/2.$

Proof (i) follows immediately from the definition of the operation of IOBC. (ii) follows by inductively applying (i), observing that g is a bit permutation, and hence a linear function, and so it distributes across the bitwise exclusive-or operation ($\oplus$). □

Remark 2. It is not hard to see that if $g^k(G_j) = G_j$ for some k, then Lemma 2(ii) could be combined with Lemma 1 to yield a forgery attack (given a ciphertext message with corresponding known plaintext). This point is made by Recacha [14], who explains that the bit permutation g has been chosen so that the smallest integer $i > 1$ such that g^i is the identity permutation is $n^2/4 - 1$. The restriction on the maximum length of messages that can be encrypted using IOBC, as defined in section 2.4, prevents this problem arising in practice. However, as we show next, in some cases g^k is 'close' to the identity permutation for significantly smaller values of k.

We conclude this section by giving certain properties of the function g. We examine two special cases of particular practical importance, i.e. where n is either 64 or 128. We first consider the case $n = 64$.

Lemma 3. *If X is a randomly selected 64-bit block then:*

$$\Pr(X = g^{341}(X)) = 2^{-22}.$$

Proof As in section 2.3, put $X = L_X||R_X$, where L_X and R_X are 31-bit and 33-bit blocks, respectively. Let $Y = g^{341}(X)$, and, analogously, let $Y = L_Y||R_Y$.

We first observe that $L_X = L_Y$. This follows immediately from the definition of g and the observation that $341 = 31 \times 11$, i.e. it is a multiple of 31.

Secondly, we show that $\Pr(R_X = R_Y) = 2^{-22}$. To establish this, first observe that $341 = 10 \times 33 + 11$, i.e. $R_Y =>_{11} (R_X)$. Since $33 = 3 \times 11$, it follows that $R_Y = R_X$ if and only if $R_X = Z||Z||Z$, where Z is an arbitrary 11-bit string. There are clearly 2^{11} such strings Z, and hence the probability that $R_Y = R_X$ is $2^{11}/2^{33}$, and the claim follows. This establishes the desired result. □

An analogous result holds for $n = 128$, as follows.

Lemma 4. *If X is a randomly selected 128-bit block then:*

$$\Pr(X = g^{1365}(X)) = 2^{-42}.$$

Proof As previously, let $X = L_X||R_X$, where L_X and R_X are 63-bit and 65-bit blocks, respectively. Put $Y = g^{1365}(X)$, and define L_Y and R_Y analogously to the proof of the previous lemma.

Since $1365 = 21 \times 65$ it follows that $R_Y = R_X$. Also, since $1365 = 21 \times 63 + 42$, and since $21|42$ and $21|63$, we have $L_Y = L_X$ if and only if $L_X = Z||Z||Z$, where Z is an arbitrary 21-bit string. The result now follows. □

Remark 3. Similar results can be achieved for any $n = 2^m$ since, for every m, either $2^m - 1$ or $2^m + 1$ is a multiple of 3.

4 A Known-Plaintext Forgery Attack on IOBC

The main elements of the attack are now in place. We suppose that the attacker has access to a number of ciphertext messages all encrypted using the same key, and that the attacker also knows large parts of the plaintext for these messages. The precise number of messages required for the attack will depend on the message lengths and the value of n. We look at two special cases of particular importance.

4.1 The Case $n = 64$

We start by considering the case $n = 64$, as applies for standardized block ciphers such as 3DES, MISTY1 and CAST-128 [17]. In this case the definition of IOBC requires that messages encrypted using IOBC contain at most 2047 blocks.

Suppose the attacker has obtained a ciphertext message $[S], C_1, C_2, \ldots, C_t$ where $t \geq 685$. Suppose also that the attacker knows the corresponding plaintext blocks $P_1, P_2, \ldots, P_t$ (in fact, the attack we describe does not require the attacker to know all the plaintext blocks, as will become clear). Using Lemma 2(ii) with $j = 1$ and $k = 341$, the attacker can use knowledge of $C_2, C_4, \ldots, C_{682}$ and $P_3, P_5, \ldots, P_{683}$ to compute $g^{341}(G_1) \oplus G_{683}$.

The attacker now constructs a new ciphertext message $[S], C_1^*, C_2^*, \ldots, C_{t-682}^*$ equal to the following sequence of blocks:

$$[S], C_1, C_{684} \oplus g^{342}(G_1) \oplus g(G_{683}), C_{685}, \ldots, C_t.$$

Note that $g^{342}(G_1) \oplus g(G_{683})$ can be obtained simply by applying g to $g^{341}(G_1) \oplus G_{683}$.

By Lemma 3, the probability that $g^{342}(G_1) \oplus g(G_{683}) = g(G_1) \oplus g(G_{683})$ is 2^{-22}, assuming that the encryption algorithm generates randomly distributed ciphertext blocks. If this event occurs, then, by Lemma 1, the result of IOBC decrypting $[S], C_1^*, C_2^*, \ldots, C_{t-682}^*$ will be equal to:

$$P_1, P_{684} \oplus F_1 \oplus F_{683}, P_{685}, P_{686}, \ldots, P_t.$$

That is, since $t \geq 685$, the final plaintext block will contain the correct ICV, i.e. $[S], C_1^*, C_2^*, \ldots, C_{t-682}^*$ will be a successful forgery.

The above attack, which essentially involves cutting out 682 consecutive ciphertext blocks from a valid message and modifying the ciphertext block immediately after the removed portion, will yield a successful forgery with probability 2^{-22}. In the example above, the removed ciphertext blocks were $C_2, C_3, \ldots, C_{683}$, but essentially the same attack will work by removing any sequence of 682 consecutive ciphertext blocks as long as it does not include the first block or the final two blocks. Thus, for example, a message containing 1808 blocks (well short of the maximum of 2047) could be used to construct $1024 = 2^{10}$ different possible forgeries, each of which would have a probability of 2^{-22} of being accepted as legitimate. A simple argument shows that 1000–2000 encrypted messages of this length could therefore yield 2^{21}–2^{22} forgeries, at least one of which is likely to be accepted.

We therefore conclude that the IOBC integrity protection mechanism can, in this case, be defeated with potentially as few as 1000–2000 known plaintexts and 2^{21}–2^{22} queries to a decrypting party.

4.2 The Case $n = 128$

We next consider the case $n = 128$, as applies for standardized block ciphers such as AES, Camellia and SEED [17]. In this case the definition of IOBC requires that messages encrypted using IOBC contain at most 8191 blocks.

An exactly analogous approach will clearly work here as for the $n = 64$ case, except that in this case we need to omit 2730 consecutive ciphertext blocks from a valid message, and make appropriate modifications to the ciphertext block immediately following the omitted sequence. In this case, the probability of the forged message being accepted will be 2^{-42} (from Lemma 4). As in the 64-bit case, a single message could yield a number of possible forgeries. For example, a 6829-block message could be used to generate 2^{12} different possible forgeries. A total of 2^{41}–2^{42} forgery attempts will be required to have a good chance of having at least one forgery accepted, potentially requiring 2^{29}–2^{30} known plaintexts (if they have an average length of around 7000 blocks).

This is a rather large number, but significantly less than the 2^{64} which is the design goal.

4.3 Other Values of n

The same general approach will work for any value of n (see Remark 3), yielding a known-plaintext-based forgery attack with complexity approximately $2^{n/3}$ decryptions and somewhat less than $2^{n/3}$ known plaintexts.

5 Can IOBC be Fixed?

It is not hard to see that the attacks in section 4 could be prevented by further limiting the maximum length of message that can be encrypted using IOBC mode. However, unless the limit is made very small, less effective versions of the attack described in section 4 will still apply, where the exact results will depend on the factorisation of $2^m - 1$ and $2^m + 1$.

For example, for the $n = 64$ case (i.e. $m = 6$) we know that, for randomly chosen blocks (X, Y) of 33 and 31 bits respectively, $\Pr(g^{93}(X) = X) = 2^{-30}$ and $g^{93}(Y) = Y$. That is a forgery attack with a success probability of 2^{-30} could be launched using a ciphertext (with known plaintext) of length only 100 blocks.

Of course, it may be possible to devise significantly more secure schemes by choosing g to be more complex, but this would reduce the attractiveness of the scheme. After all, the only reason to adopt this approach instead of ‘encrypt then MAC’ (which is provably secure) is to reduce the complexity of protecting the message to that of encryption plus a small delta.

6 A Chosen-Plaintext Forgery Attack

All the attacks we have considered so far can be avoided if only relatively short messages are encrypted. Moreover, these attacks take advantage of special properties of the function g. As a result, it is of at least theoretical interest to know the level of security provided by IOBC mode regardless of the length of plaintext messages and of the choice of g.

We thus conclude the main part of the paper by sketching a certificational chosen-plaintext-based forgery attack which serves to limit the security of IOBC regardless of length limits for plaintexts (and the choice of g). Suppose that $[S], C_1, C_2, \ldots, C_t$ and $[S'], C'_1, C'_2, \ldots, C'_{t'}$ are IOBC encrypted versions of the plaintext sequences $P_1, P_2, \ldots, P_t$ and $P'_1, P'_2, \ldots, P'_{t'}$, respectively. Suppose also that $P'_i = P_j$ and $P'_{i+1} = P_{j+1}$.

It is not hard to see that if $C'_i = C_j$ and $C'_{i+1} = C_{j+1}$ then, with very high probability, we have $F'_{i-1} = F_{j-1}$, and hence $G'_{i-1} = G_{j-1}$ and $G'_{i+1} = G_{j+1}$. If such an event occurs, then, by Lemma 1, the constructed ciphertext message $[S'], C'_1, C'_2, \ldots, C'_{i-1}, C_j, C_{j+1}, \ldots, C_t$ will very conveniently decrypt to $P'_1, P'_2, \ldots, P'_{i-1}, P_j, P_{j+1}, \ldots, P_t$, i.e. a MAC forgery has been constructed. By the usual 'birthday paradox' probabilistic arguments, to find such an event simply requires around $2^{n/2}$ chosen plaintexts to be encrypted, each containing the same consecutive pair of plaintext blocks. In fact, the number of required chosen plaintext encryptions can be reduced to significantly less than $2^{n/2}$ by including many occurrences of the fixed pair of plaintext blocks in each chosen plaintext.

That is, regardless of the lengths of plaintext messages and the choice of g, forgery attacks on IOBC are possible if of the order of $2^{n/2}$ messages are encrypted using the same key.

7 Summary and Conclusions

The analysis in this paper suggests that IOBC does not offer an adequate level of security for routine use as the basis of a combined integrity and confidentiality technique. In fact, use of the 'add redundancy and then encrypt using a special mode' approach to provide combined integrity and confidentiality protection is no longer 'state of the art', and so this is arguably not a major development. The main significance is that, as mentioned in section 1, IOBC was the only remaining proposed block cipher mode for simultaneous confidentiality and integrity protection known to the author which had not already been shown to suffer from forgery attack issues. Hence this paper serves to bring a cryptographic chapter to a tidy close.

As discussed in many other places, if both confidentiality and integrity protection are required, then either encryption and a MAC should be combined in an appropriate way, or a dedicated 'authenticated encryption' mode should be used — see, for example, ISO/IEC 19772 [3]. Indeed, a wide variety of provably secure schemes are available.

Acknowledgements. The author would like to thank Francisco Recacha for his very helpful explanations and corrections, and also the anonymous referees for comments which have helped to improve the presentation of this paper.

References

1. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1997)
2. McGrew, D.A., Viega, J.: The Galois/Counter mode of operation (GCM). Available at: `http://www.mindspring.com/~dmcgrew/gcm-nist-6.pdf` (May 2005)
3. International Organization for Standardization Genève, Switzerland: ISO/IEC 19772:2009, Information technology — Security techniques — Authenticated encryption mechanisms. (February 2009)
4. International Organization for Standardization Genève, Switzerland: ISO/IEC 9797-3:2011, Information technology — Security techniques — Message Authentication Codes (MACs) — Part 3: Mechanisms using a universal hash-function. (2011)
5. Dent, A.W., Mitchell, C.J.: User's Guide to Cryptography and Standards. Artech House (2005)
6. Kohl, J.T.: The use of encryption in Kerberos for network authentication. In Brassard, G., ed.: Advances in Cryptology — CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20–24, 1989, Proceedings. Volume 435 of Lecture Notes in Computer Science., Springer-Verlag, Berlin (1990) 35–43
7. Steiner, J., Neuman, C., Schiller, J.: Kerberos: an authentication service for open network systems. In: Proceedings: Usenix Association, Winter Conference, Dallas 1988, USENIX Association, Berkeley, California (February 1988) 191–202
8. Meyer, C.H., Matyas, S.M.: Cryptography: A new dimension in computer data security. John Wiley and Sons, New York (1982)
9. Mitchell, C.J.: Cryptanalysis of two variants of PCBC mode when used for message integrity. In Boyd, C., Gonzalez Nieto, J.M., eds.: Information Security and Privacy, 10th Australasian Conference, ACISP 2005, Brisbane, Australia, July 4–6 2005, Proceedings. Number 3574 in Lecture Notes in Computer Science, Springer-Verlag, Berlin (2005) 560–571
10. Sierra, J.M., Hernandez, J.C., Jayaram, N., Ribagorda, A.: Low computational cost integrity for block ciphers. Future Generation Computer Systems **20** (2004) 857–863
11. Gligor, V.G., Donescu, P.: Integrity-aware PCBC encryption schemes. In: Security Protocols, 7th International Workshop, Cambridge, UK, April 19-21, 1999, Proceedings. Volume 1796 of Lecture Notes in Computer Science., Springer-Verlag, Berlin (2000) 153–171
12. Ferguson, N., Whiting, D., Kelsey, J., Wagner, D.: Critical weaknesses of iaPCBC. (November 1999)
13. Zuquete, A., Guedes, P.: Transparent authentication and confidentiality for stream sockets. IEEE Micro **16**(3) (May/June 1996) 34–41
14. Recacha, F.: IOBC: Un nuevo modo de encadenamiento para cifrado en bloque. In: Proceedings: IV Reunion Espanola de Criptologia, Valladolid, September 1996. (1996) 85–92

15. Zuquete, A., Guedes, P.: Efficient error-propagating block chaining. In Darnell, M., ed.: Cryptography and Coding, 6th IMA International Conference, Cirencester, UK, December 17–19, 1997, Proceedings. Number 1355 in Lecture Notes in Computer Science, Springer-Verlag, Berlin (1997) 323–334
16. Mitchell, C.J.: Cryptanalysis of the EPBC authenticated encryption mode. In Galbraith, S.D., ed.: Cryptography and Coding, 11th IMA International Conference, Cirencester, UK, December 18-20, 2007, Proceedings. Volume 4887 of Lecture Notes in Computer Science., Springer-Verlag, Berlin (2007) 118–128
17. International Organization for Standardization Genève, Switzerland: ISO/IEC 18033-3:2010, Information technology — Security techniques — Encryption algorithms — Part 3: Block ciphers. 2nd edn. (2010)