Error Oracle Attacks on CBC Mode: Is There a Future for CBC Mode Encryption?

Chris J. Mitchell

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

Abstract. This paper is primarily concerned with the CBC block cipher mode. The impact on the usability of this mode of recently proposed padding oracle attacks, together with other related attacks described in this paper, is considered. For applications where unauthenticated encryption is required, the use of CBC mode is compared with its major symmetric rival, namely the stream cipher. It is argued that, where possible, authenticated encryption should be used, and, where this is not possible, a stream cipher would appear to be a superior choice. This raises a major question mark over the future use of CBC mode, except as part of a more complex mode designed to provide authenticated encryption.

1 Introduction

The CBC (Cipher Block Chaining) 'mode of operation' for a block cipher has been in wide use for many years. A mode in this sense is simply a way of using a block cipher to encrypt a string of bits (often referred to as a 'message').

CBC mode, as originally specified in the 1980 US FIPS Pub. 81 [1], was first defined as one of four modes of use for the DES block cipher [2]. Since then, CBC mode, together with the other three modes from FIPS 81, has appeared in a number of other standards, including ISO/IEC 10116, the international standard for modes of operation (the second edition of which was published in 1997 [3], and a third edition of which is nearing completion [4]). For further details of block cipher modes of operation see, for example, Chapter 5 of [5].

2 Encryption and integrity-protection

CBC mode, along with all the other modes of operation standardised in ISO/IEC 10116, is designed only to provide *confidentiality* protection for encrypted data. Thus, if the integrity and/or origin of the data is also to be protected, then use of a separate mechanism, e.g. a Message Authentication Code (MAC) or a digital signature is required; see, for example, [5, 6] for discussions of these cryptographic primitives and for details of relevant standards.

Over the last few years, a number of proposals for new modes of operation offering both confidentiality and integrity protection have appeared. These

modes, often referred to as 'authenticated-encryption techniques', include OCB [7], EAX [8] and CCM [9, 10]. These techniques are also currently being standardised — the second working draft of what is intended to become ISO/IEC 19772 on authenticated encryption was published late in 2004 [11].

In parallel with these recent developments, a number of implementation-based attacks against CBC mode have been discovered — see, for example, [12–16]. In these attacks, use of a so called 'padding oracle' enables an attacker to discover information about the plaintext for a CBC-encrypted message. More specifically, we suppose that the decrypting device, after recovering the plaintext from the ciphertext, checks that the padding format is correct. If it is not, an error message is generated, the presence or absence of which can be detected by the cryptanalyst. This constitutes the 'padding oracle', and practical examples of the existence of such oracles has been demonstrated. The cryptanalyst uses such an oracle by making carefully designed modifications to ciphertexts, and then observing whether or not the modified ciphertext induces a padding failure — this, in turn, reveals information about the plaintext.

There are two main responses to the existence of such attacks, which appear to pose a genuine threat to the security of some secure communications systems. (As we discuss below, not all systems are subject to such attacks; however, the possibility of such attacks may be sufficiently significant to mean that adopting countermeasures across the board is probably advisable).

- The first is to observe that error messages of all kinds, including padding error messages, should be designed with care. Careful implementation of such messages would probably have prevented the practical realisation of most, if not all, of the so far described attacks.
- The second, most notably advocated by Black and Urtubia [12], and also by Paterson and Yau [14], is to always provide integrity in conjunction with encryption, and to arrange error messages appropriately. Clearly, for such an approach to be effective, the integrity check must be performed before any necessary padding is checked. If this line of argument is followed, then the most logical approach is to use an authenticated-encryption technique such as one of those referred to above.

The second of the above arguments is clearly convincing, and is one we return to below in suggesting that CBC mode should never be used without some accompanying integrity check. However, for practical reasons we do not support the argument that encryption should never be used without an accompanying integrity check. The reason for this latter claim is that there appear to be applications where unauthenticated encryption is needed. These include the following.

- *Applications where data errors are acceptable.* If the data to be encrypted consists of image or audio data (e.g. a digitised voice or video channel), then a certain proportion of errors in the recovered plaintext data may be acceptable to the recipient. This is because, after conversion back to an analogue version, the resulting (corrupted) signal will still be usable. For example, a modest number of errors in a digitised voice signal will often

result in a degraded but nevertheless comprehensible version. Moreover, if the communications system in use required all such corrupted signals to be rejected, retransmission may not be an option, e.g. for a real-time audio or video channel (as would be used in a telephone call or video conference). In such a case, a slightly corrupted version of the original signal is clearly preferable to no signal at all.
Hence, if an integrity check is used in such a scenario, the result will be an unacceptable degradation in the channel. Thus, in these circumstances (as arise, for example, in mobile telephone wireless transmissions) use of a cryptographic integrity check is not really practical. Current such applications typically use a stream cipher because of its lack of error propagation.
Of course, use of an error-correcting code applied to the entire ciphertext may alleviate such problems and allow use of an authenticated encryption mode. However, if the error rates are highly variable, then such an approach may simply be too complex to be practicable (and any scheme that imposes latency will be unacceptable in real-time applications, such as voice).

– *Very high bandwidth channels (bulk encryption).* The second case is where very large volumes of data are to be encrypted at high speed, for example, when encrypting all of the data sent on a high bandwidth channel, such as an optical fibre trunk. One major advantage of encrypting at a low level of the protocol hierarchy is that all address information can be encrypted, revealing no information about traffic flows to an interceptor.
In this case it may simply be impractical to include an integrity check, typically because generating and verifying such values, and dealing with any necessary retransmissions, at very high data rates may be infeasibly complex. It is arguably more appropriate to provide error protection at higher levels of the protocol hierarchy.

As a result of these and other applications of unauthenticated encryption, we claim that mandating authenticated encryption is not always possible. As a result it is necessary to decide which types of encryption are most appropriate when integrity checks are not performed, and this is the main theme of this paper.

Finally note that trivial distinguishing attacks exist on CBC in a chosen ciphertext setting. The main contribution of this paper, and the earlier work on padding oracle attacks, is to demonstrate that one can also perform message recovery attacks, which are, of course, stronger than distinguishing attacks.

3 CBC mode — definition, properties, and a fundamental observation

We next describe how CBC mode works, and outline important properties.

3.1 Definition of CBC mode

Use of CBC mode encryption requires that the plaintext to be encrypted is first padded so that its length is a multiple of n bits, where n is the block length

of the block cipher in use. The padded plaintext is then divided into a series of n-bit blocks: $P_1, P_2, \ldots, P_q$, say. An n-bit *starting variable* (also sometimes called an *initialisation vector* or IV) is also required.

If the chosen starting variable is denoted by S, then encryption involves computing a sequence of ciphertext blocks $C_1, C_2, \ldots, C_q$, as follows:

$$C_1 = e_K(P_1 \oplus S), \quad C_i = e_K(P_i \oplus C_{i-1}), \ \ (i > 1)$$

where $e_K(X)$ denotes the block cipher encryption of n-bit block X using the secret key K, and $\oplus$ denotes the bit-wise exclusive-or of blocks.

3.2 Properties of CBC mode

In CBC mode, if the same message is enciphered twice then the same ciphertext will result, unless the starting variable is changed. Moreover, if two messages agree for the first t blocks, for some t, then the first t blocks of ciphertext will be the same (again unless a different starting variable is used). Hence the starting variable S should be different for every message.

A 'proof of security' of CBC mode was published by Bellare et al. in 1997 [17]. This proof requires the starting variable S to be random and not selectable by an attacker; in fact there are also advantages with choosing S to be a secret (known only to the legitimate sender and receiver). This is supported by recent work of Rogaway [18], who obtains superior security proofs for this technique when the starting variable is a one-time secret.

Managing starting variables is clearly a non-trivial issue for the user. One way of achieving the use of a different value of S for every encrypted message is simply to generate a random value for S, and to send this with the encrypted message. However this does not meet the requirement that starting variables should ideally be secret. Providing a different secret starting variable for every message can be achieved in a variety of ways, including sending a counter with the message and using an encrypted version of this counter as the starting variable, or generating a random value for every message and encrypting it before sending it to the recipient with the encrypted message.

Use of CBC mode results in a property known as *error propagation*. That is, a single bit error in the ciphertext will result in the loss of an entire block of plaintext. Moreover, the corresponding single bit in the next plaintext block will also be in error. To see why this holds, consider the decryption step used to yield P_i (for any i), namely: $P_i = d_K(C_i) \oplus C_{i-1}$, where d denotes block cipher decryption. First observe that P_i is a function of just two ciphertext blocks: C_i and C_{i-1}. Also, if C_i contains one or more bit errors, then P_i will be completely garbled because of the randomising effects of the block cipher. Finally, if C_{i-1} contains one bit error, then this will affect the recovered value of P_i in precisely the same bit position.

3.3 A key observation

We next point out a simple yet important property of CBC mode that gives rise to both padding oracle attacks and more general message-content based attacks on this mode of operation.

Suppose $P_1, P_2, \ldots, P_q$ is a (padded) plaintext message which has been CBC-encrypted to obtain the ciphertext $C_1, C_2 \ldots, C_q$, using the block cipher secret key K and the starting variable S. Suppose also that a cryptanalyst submits a ciphertext $X_1, X_2, \ldots, X_{s-1}, C_j, X_{s+1}, \ldots, X_t$ for decryption, where $1 < s \leq t$ and $j > 1$, and that the decrypted result is $P'_1, P'_2, \ldots, P'_t$.

Then $P'_s = d_K(C_j) \oplus X_{s-1}$ (regardless of which starting variable is used in the decryption, since $s > 1$). Moreover, by definition, $P_j = d_K(C_j) \oplus C_{j-1}$ (since $j > 1$). Hence we have the following simple equation:

$$P'_s \oplus P_j = X_{s-1} \oplus C_{j-1}. \tag{1}$$

This equation is the basis of all the padding oracle attacks referred to above. It is also the reason why we question here the use of CBC mode without any accompanying data integrity check. More specifically, equation (1) is the basis of two main types of attack designed to learn information about the plaintext corresponding to an encrypted message. These are as follows.

1. The first class of attack is designed to learn information about a single block of plaintext. Using the above notation, the cryptanalyst sets $X_{s-1} = C_{j-1} \oplus Q$ where Q is a particular bit pattern (e.g. containing just a single '1' bit in a chosen position); the other values X_i can be chosen arbitrarily. Then, from (1), we immediately have:

   $$P'_s \oplus P_j = Q. \tag{2}$$

   That is, the attacker can select the exact difference between P_j and the plaintext block P'_s obtained by the decrypter. If the attacker also has a means of learning whether or not the recovered plaintext block P'_s generates some type of formatting error, then this approach will enable the attacker to learn precisely targetted information about the plaintext block P_j.
2. The second class of attack involves learning information about a pair of consecutive plaintext blocks for an enciphered message $C^*_1, C^*_2, \ldots, C^*_t$ (which may or may not be be the same as $C_1, C_2, \ldots, C_q$, although it must have been encrypted using the same block cipher key K). Suppose that the $P^*_1, P^*_2, \ldots, P^*_t$ is the plaintext corresponding to ciphertext $C^*_1, C^*_2, \ldots, C^*_t$. Using the previously established notation, the cryptanalyst sets $X_i = C^*_i$ $(i \neq s)$ and submits the resulting ciphertext to the decrypter.
   Note that we are here concerned with the entire plaintext message, and so we need to consider which starting variable will be used by the decrypter to recover the plaintext. For the purposes of discussing this case we assume that the starting variable is always sent with the ciphertext, perhaps in encrypted form. As a result the attacker has some control over the starting variable; in

particular the attacker can ensure that the starting variable originally used to encrypt the ciphertext $C_1^*, C_2^*, \ldots, C_t^*$ is used on each occasion.
Then, applying (1), we immediately have:

$$P'_i = P_i^*, \ \ (i \neq s;\ i \neq s+1) \tag{3}$$

$$P'_s \oplus P_j = C_{s-1}^* \oplus C_{j-1}, \ \text{and} \tag{4}$$

$$P'_{s+1} \oplus P_{s+1}^* = C_s^* \oplus C_j. \tag{5}$$

In this case the attacker will therefore know that the plaintext message $P_1^*, P_2^*, \ldots, P_t^*$ and the message $P'_1, P'_2, \ldots, P'_t$ recovered by the decrypter will be identical in all blocks except for numbers s and $s+1$, where we have:

$$P'_s \oplus P_s^* = P_s^* \oplus P_j \oplus C_{s-1}^* \oplus C_{j-1}, \ \text{and} \tag{6}$$

$$P'_{s+1} \oplus P_{s+1}^* = C_s^* \oplus C_j. \tag{7}$$

If the attacker has a means of learning whether or not the recovered plaintext will generate some type of formatting error, then this approach will potentially enable the attacker to learn information about $P_s^* \oplus P_j$. This will arise if the difference between two correctly formatted messages always possesses a certain property. We give an example of such an attack below.

4 Error oracle attacks

The idea behind a padding oracle attack was outlined in Section 2. In such an attack it is assumed that the attacker has one or more valid ciphertexts, and can also inject modified ciphertexts into the communications channel. Moreover, the decrypter will, immediately after decryption, check that the padding employed in the recovered plaintext is in the correct format or not. If it is not, the decrypter is assumed to generate an error message which can be detected by the attacker — whether or not an error message is generated provides the 'padding oracle', which can be used to learn information about a message.

We now consider what we call an *error oracle* attack. In this scenario an attacker, as for a padding oracle attack, submits an encrypted message to a decrypter. The decrypter expects all plaintext messages to contain certain structure, and we suppose that the nature of this structure is known to the attacker. We further suppose that, in the absence of such structure, the decrypter exhibits behaviour different to that it exhibits if the structure is present, and that this behaviour is detectable by the attacker. Examples of possible detectable behaviours include the sending of an error message or the failure to carry out an action, e.g. sending a response. The attacker then submits carefully tailored ciphertext messages to the decrypter, and thereby learns information about the plaintext from the behaviour of the decrypter. Padding oracles are simply a special case of these error oracles. Note that an error oracle is very similar to what Bellare, Kohno and Namprempre [19] refer to as a *reaction attack*. More generally, these are all examples of what have become known as *side channel attacks*.

Whilst the possibility of such attacks has been practically demonstrated, such oracles will not always exist. Indeed, such oracle attacks will probably only be possible in certain special circumstances. It is thus possible to argue that selection of cryptographic techniques should only take account of such attacks in circumstances where they are likely to arise. The problem with this is that, when designing a cryptographic protocol, it is not easy to predict when implementations might be subject to error oracle attacks. Indeed, the error oracle may exist in a higher level protocol, designed and implemented completely independently of the cryptographic functionality. We thus suggest that it is good practice always to design cryptographic schemes such that error oracles are never a threat, and we make this assumption throughout the remainder of this paper.

We next give three examples of how error attacks might be realised in practice. In each case we suppose that an attacker has intercepted a CBC-encrypted ciphertext $C_1, C_2, \ldots, C_q$ (the *target ciphertext*) for which as much information as possible is to be obtained about the corresponding (padded) plaintext $P_1, P_2, \ldots, P_q$ (the *target plaintext*).

Before proceeding note that in the first example we need the attacker to be able to force the decrypter to re-use the starting variable originally used to encrypt the message. However, the other two attacks work regardless of which starting variable the decrypter uses.

4.1 Example 1: A linear error detection attack

Suppose that a higher-level protocol is designed to error-protect all the messages it sends. Suppose further that the technique used for this error-protection is a 16-bit CRC (Cyclic Redundancy Check). We thus suppose that the target plaintext $P_1, P_2, \ldots, P_q$ incorporates a 16-bit CRC. This is, of course, bad practice, but it might be mandated by a higher level protocol designed completely independently of the protocol responsible for data encryption. Suppose also that the attacker can find out, for any chosen ciphertext, whether or not the error detection process fails after decryption (this is our error oracle).

Next suppose that the attacker constructs a query to the error oracle by replacing ciphertext block C_s with C_j for some $s \neq j$ ($s > 1$, $j > 1$) in the ciphertext string $C_1, C_2, \ldots, C_q$ (the attacker also arranges for the decrypter to use the same starting variable as was originally used to produce $C_1, C_2, \ldots, C_q$). If the 'plaintext' recovered by the decrypter is labelled $P'_1, P'_2, \ldots, P'_q$, then, from equations (6) and (7), we immediately have:

$$\begin{aligned} P'_i \oplus P_i &= 0, \ \ (1 \leq i < s \text{ and } s+1 < i \leq q), \\ P'_s \oplus P_s &= P_s \oplus P_j \oplus C_{s-1} \oplus C_{j-1}, \ \text{ and} \\ P'_{s+1} \oplus P_{s+1} &= C_s \oplus C_j. \end{aligned}$$

Given that the original message contains a CRC check, the corrupted plaintext will contain a valid CRC if and only if the ex-or of the valid message with the corrupted message has a valid CRC (by linearity). Moreover, from the above

equations the attacker knows precisely the form of this exclusive-or, with the only unknown being the value of $P_s \oplus P_j$. The probability that the corrupted message will pass the CRC is only 2^{-16}, but in this event the attacker will essentially know 16 bits of information about $P_s \oplus P_j$, since we will know that a degree 16 polynomial divides a polynomial with coefficients involving $P_s \oplus P_j$ and some known values.

Hence after an expected number of around 2^{15} CRC error oracle queries we will have learnt at least 16 bits of information about the message. A message containing $2^8 = 256$ n-bit blocks will have nearly 2^{16} candidate ordered pairs (s, j), i.e. there is a good chance that at least one of the 'corrupted' messages will yield a correct CRC. Given that a sufficient number of different error oracle queries can be constructed, this technique can be used to discover up to $16(q-2)$ bits of information regarding the plaintext $P_1, P_2, \ldots, P_q$.

This general approach can be extended in several ways. First, note that the ciphertext $C_1, C_2, \ldots, C_q$ could be modified by replacing more than block, giving more possible variants to be submitted to the error oracle. Second, the replacement ciphertext block could be taken from a different encrypted message (as long as it has been encrypted using the same key). Third, the same approach will work if the message contains any other type of error protection based on a linear code. If, for example, an 8-bit CRC was used instead of a 16-bit CRC, then discovering 8 bits of information about the plaintext would require an expected number of only around 128 queries.

4.2 Example 2: A message structure attack

For our second example we suppose that the target plaintext $P_1, P_2, \ldots, P_q$ contains a fixed byte in a known position. Suppose that the fixed byte is the kth byte in block P_s for some $s > 1$. There are many protocols that set certain bytes to zero (or some other fixed pattern) as 'future proofing', e.g. to enable the recipient of a message to determine which version of a protocol is being used. Suppose also that if this particular byte of a decrypted message is not set to the expected value then the decrypter will exhibit a particular detectable behaviour.

This scenario enables the attacker to learn the value of the kth byte of all but the first block of the plaintext using a series of error oracle queries, the expected number of which will be around 128 per block, as follows. For each j ($1 < j \leq q$; $j \neq s$), the attacker constructs a series of 'ciphertexts' with modifications to just two blocks C_{s-1} and C_s, where the modified ciphertext has the form:

$$C_1, C_2, \ldots, C_{s-2}, C_{j-1} \oplus Q_t, C_j, C_{s+1}, C_{s+2}, \ldots, C_q$$

for $t = 0, 1, \ldots, 255$. The n-bit block Q_t has as its kth byte the 1-byte binary representation of t, and zeros elsewhere. The attacker submits these ciphertexts to the error oracle in turn, until one is found which does not cause an error, i.e. the recovered plaintext $P'_1, P'_2, \ldots, P'_q$ for the manipulated ciphertext has the property that the kth byte of P'_s is equal to the correct fixed byte. If this occurs, say, for Q_u, then, from equation (2), the attacker immediately knows that

$$P_j = P'_s \oplus Q_u.$$

That is, given that the kth byte of P'_s is known to equal the fixed byte, the attacker has discovered the value of the jth byte of P_j. This approach can be used to find the kth byte of every block of the original plaintext (except for P_1).

Similar results hold for parts of bytes or multiple bytes.

4.3 Example 3: Content-based padding oracle attacks

The third attack we consider is a type of padding attack which will only work if the attacker knows something about the message structure (and this structure has appropriate properties). This differs from a 'standard' padding oracle attack which does not require any assumptions to be made regarding the plaintext. However, such a scenario is not particularly unlikely — it also enables us to attack padding methods which are essentially immune to regular padding oracle attacks.

First suppose that the CBC-encrypted data is a fixed length message, and that the attacker knows the message length, which we suppose is equal to $(q-1)n+r$ (where q and r satisfy $q \geq 1$ and $1 \leq r < n$). Suppose, moreover, that padding method 1 from ISO/IEC 9797-1 [20] is in use; that is, suppose that padding merely involves adding zeros to the end of the message until the message length is a multiple of n bits[1]. Hence the attacker knows that the last $n-d$ bits of P_q are all zeros.

This scenario enables the attacker to learn the value of the last $n-d$ bits of all but the first block of the plaintext, using an expected number of around 2^{n-d-1} error oracle queries per block. For each j $(1 < j \leq q;\ j \neq 1)$, the attacker constructs a series of 'ciphertexts' with modifications to the final two blocks C_{q-1} and C_q, where the modified ciphertext has the form:

$$C_1, C_2, \ldots, C_{q-2}, C_{j-1} \oplus Q_t, C_j$$

for $t = 0, 1, \ldots, 2^{n-d}-1$. The n-bit block Q_t has as its final $n-d$ bits the binary representation of t, and zeros elsewhere. The attacker submits these ciphertexts to the error oracle in turn, until one is found which does not cause an error, i.e. the recovered plaintext $P'_1, P'_2, \ldots, P'_q$ for the manipulated ciphertext has the property that the final $n-d$ bits of P'_q are all zeros. If this occurs for Q_u say, then, from equation (2), the attacker immediately knows that

$$P_j = P'_q \oplus Q_u.$$

That is, given that the final $n-d$ bits of P'_q are known to be all zeros, the attacker has discovered the value of the final $n-d$ bits of P_j. This approach can be used to find the final $n-d$ bits of every block of the original plaintext (except for P_1).

Note that such an attack would apply equally well to messages padded using padding method 2 of ISO/IEC 9797-1 [20], i.e. the method that involves adding a single one to the end of the message followed by the minimum number of zeros necessary to ensure that the padded message length is a multiple of n.

[1] Note that this padding method is only usable in circumstances where the message length is fixed.

5 Error oracle attacks on stream ciphers

So far we have focussed on CBC mode. However, one of the main objectives is to consider which method of symmetric encryption is most suited for use in circumstances where authenticated encryption is not appropriate. We therefore need to consider the vulnerability of stream ciphers to error oracle attacks, since stream ciphers are the main alternative to use of CBC mode. Note that by stream ciphers we mean to include use of a block cipher in CTR and OFB modes.

First, observe that stream ciphers typically do not require the use of padding, and hence padding oracle attacks are not an issue. Black and Urtubia [12] point out that, on occasion, stream ciphers do use padding, although it is not clear how often this occurs; moreover, a best practice recommendation to never pad plaintext prior to use of a stream cipher could eliminate any such issues.

Second, we claim that error oracle attacks analogous to those based on equations (6) and (7) do not apply for stream ciphers, since, when using a stream cipher, different parts of a single ciphertext message are encrypted using different keystream sequences; hence it is not possible to learn anything about the plaintext by exoring two different portions of ciphertext. The same is true when combining two different ciphertexts since, even if the ciphertext strings are taken from the same point in the encrypted messages, different keystream sequences will be used (as long as starting variables are employed to ensure that different messages are encrypted using different keystream sequences).

Third, observe, however, that error oracle attacks analogous to those based on equation (2) *do* apply to stream ciphers. This arises because a single bit change in stream cipher ciphertext gives rise to a single bit change in the same position in the recovered plaintext. We consider a simple, but not necessarily unrealistic, example. Suppose that an attacker knows that two consecutive plaintext bits will always be equal to one of three possibilities, namely: 00, 01 and 10. Suppose, moreover, that the combination 11 will cause a formatting error detectable by an attacker. If the ciphertext bit corresponding to the second of these 'formatting' bits is changed, and the resulting ciphertext is submitted to the error oracle, then if there is no error then the attacker knows that the first plaintext bit of the two is a zero, and if there is an error then the attacker knows that the first plaintext bit of the two is a one.

In summary, although stream ciphers are certainly not immune to error oracle attacks, the risk is somewhat less serious than for CBC mode, since less attack variants apply in this case. Also note that, although a recently proposed attack on the GSM stream cipher uses the fact that the plaintext that is stream ciphered is redundant [21], the main problem arises because of the relatively weak keystream generator in use, not through padding oracle attacks.

6 CBC mode versus stream ciphers

We now consider whether a stream cipher or CBC mode encryption is more suitable for use in cases where authenticated encryption is not appropriate. We start by considering the impact of error oracle attacks.

The recent focus by a number of authors on padding oracle attacks has led to the impression that problems can be addressed by either managing padding error messages more carefully or (preferably) by choosing a padding method which cannot be exploited. An obvious candidate for such a technique is padding method 2 from ISO/IEC 9797-1 [20], i.e. the method that involves adding a single one followed by the minimum necessary number of zeros. However we should point out that Black and Urtubia [12] do point out some residual issues with this technique, although they would appear to be much less serious than the issues for other padding methods. Black and Urtubia also propose other padding methods for which padding oracle attacks cannot succeed.

However, the content-based padding oracle attack described in Section 4.3 suggests that no padding method is 'safe' when an attacker knows information about the structure of the message and has access to an error oracle. Moreover, simply requiring that systems should be designed not to give error oracles is not realistic. This is because the error oracle may be part of a higher-level protocol, designed completely independently of the protocol layer implementing encryption. That is, the presence of such error oracles may be something out of the hands of the designer and implementer of the encryption system.

We next observe that, as discussed in Section 5, CBC mode encryption is at a significantly greater risk from error oracle attacks than stream cipher encryption. This is because use of a stream cipher typically involves no padding, and only some error oracle attacks work.

This suggests the following preliminary conclusions, namely that: (a) authenticated encryption should be used wherever possible, and (b) if unauthenticated encryption is necessary, then stream ciphers appear to offer certain advantages over CBC mode with reference to side channel attacks. We next looks at how these preliminary findings need to be modified in the context of the two example cases where unauthenticated encryption is appropriate (as discussed in Section 2).

- *Applications where data errors are acceptable.* In such an application it is very important that the encryption technique does not significantly increase the error rate. That is, if the channel has the property that the error probability for a received ciphertext bit is p, then the probability of an error in a plaintext bit after decryption should not be significantly greater than p. This property holds for a stream cipher, but does not hold for CBC mode, where the error probability will be increased from p to around $(n/2+1)p$ (for small p). Hence, in this type of application, as exemplified by the choice of a stream cipher for GSM and UMTS encryption, a stream cipher has very significant advantages over CBC mode.
- *Very high bandwidth channels (bulk encryption).* Here it is important that the cipher be capable of running at the highest possible speed (for a given complexity of hardware). Typically, stream ciphers, such as SNOW 2.0 [22] or MUGI [23], can be implemented to run significantly faster than CBC-mode block cipher encryption. Hence again stream ciphers offer significant practical advantages.

7 Conclusions: The end of CBC mode?

As we have mentioned above, the existing discussions of padding oracle attacks give the impression that the error oracle problem can be solved by designing padding methods appropriately and ensuring that padding error messages are carefully designed. Whilst there is no doubt that, if CBC mode it to be used, then it should be used with a carefully selected padding method[2], this by no means solves all the issues associated with error oracles.

However, we would suggest that the problem is more general than this. As we have demonstrated, if messages to be encrypted contain certain types of known structure, then error oracle attacks may be possible regardless of the padding method used. Moreover, the designer of the encryption protocol cannot always predict the nature of the messages that are to be protected using the protocol, and hence preventing such attacks by stopping structured messages is essentially impossible. As we have already pointed out, this problem is known to arise elsewhere, as exemplified by certain attacks on GSM encryption [21].

Whilst all these problems would be avoided if the encryption protocol provided both confidentiality and integrity checking, we have shown that this is not always appropriate. Thus the designer of an symmetric encryption system for which it is not appropriate to provide integrity protection is typically faced with a choice between CBC mode encryption and use of a stream cipher. We suggest that a stream cipher is always to be preferred for two main reasons: first, stream ciphers are less prone to error oracle attacks (although not completely immune), and second, they appear to be a much better fit to those particular applications where it is not appropriate to provide integrity checking. These considerations apply despite the fact that stream ciphers are 'IV sensitive', i.e. re-use of an IV for a stream cipher is very dangerous.

Hence, as a result, for any system employing symmetric encryption, the choice would appear to be between a combination of symmetric encryption of some kind and an integrity check (such as a MAC) or a stream cipher (including use of a block cipher in CTR or OFB modes). However, as argued by a number of authors (see, for example, Bellare, Kohno and Namprempre [19]) it is important to combine encryption and authentication with care to avoid unintended weaknesses. This suggests that it is probably always desirable to use a specifically designed authenticated-encryption mode (some of which also have efficiency advantages), rather than an ad hoc combination of encryption and a MAC.

Thus our conclusion is that there would appear to be two main choices for the user of a symmetric encryption system: an authenticated-encryption system (see, e.g. [7–10, 24]) or a stream cipher. (Of course, there do exist other possibilities, including the use of all-or-nothing transforms, introduced by Rivest [25], and modes based on tweakable block ciphers [26] included in draft standards produced by the IEEE Security in Storage Working Group — see `siswg.org`).

[2] This observation has influenced the UK ballot comments on ISO/IEC FCD 10116 [4], in which it is suggested that the revised standard recommends the use of Padding Method 2 from ISO/IEC 9797-1 [20].

This prompts the suggestion in the title of this paper that, except for legacy applications, naive CBC encryption should never be used, regardless of which padding method is employed.

Acknowledgements

The author would like to thank Kenny Paterson and an anonymous reviewer for a number of important helpful comments and suggestions.

References

1. National Institute of Standards and Technology (NIST) Gaithersburg, MD: Federal Information Processing Standards Publication 81 (FIPS PUB 81): DES Modes of Operation. (1980)
2. National Institute of Standards and Technology (NIST) Gaithersburg, MD: Federal Information Processing Standards Publication 46-3 (FIPS PUB 46-3): Data Encryption Standard. (1999)
3. International Organization for Standardization Genève, Switzerland: ISO/IEC 10116: 1997, Information technology — Security techniques — Modes of operation for an n-bit block cipher. 2nd edn. (1997)
4. International Organization for Standardization Genève, Switzerland: ISO/IEC FCD 10116, Information technology — Security techniques — Modes of operation for an n-bit block cipher. 3rd edn. (2004)
5. Dent, A.W., Mitchell, C.J.: User's Guide to Cryptography and Standards. Artech House (2005)
6. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1997)
7. Rogaway, P., Bellare, M., Black, J.: OCB: A block-cipher mode of operation for efficient authenticated encryption. ACM Transactions on Information and System Security **6** (2003) 365–403
8. Bellare, M., Rogaway, P., Wagner, D.: The EAX mode of operation. In Roy, B., Meier, W., eds.: Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers. Volume 3017 of Lecture Notes in Computer Science., Springer-Verlag, Berlin (2004) 389–407
9. National Institute of Standards and Technology (NIST): NIST Special Publication 800-38C, Draft Recommendation for Block Cipher Modes of Operation: The CCM Mode For Authentication and Confidentiality. (2003)
10. Whiting, D., Housley, R., Ferguson, N.: RFC 3610, Counter with CBC-MAC (CCM). Internet Engineering Task Force. (2003)
11. International Organization for Standardization Genève, Switzerland: ISO/IEC 2nd WD 19772: 2004, Information technology — Security techniques — Authenticated encryption mechanisms. (2004)
12. Black, J., Urtubia, H.: Side-channel attacks on symmetric encryption schemes: The case for authenticated encryption. In: Proceedings of the 11th USENIX Security Symposium, San Francisco, CA, USA, August 5-9, 2002, USENIX (2002) 327–338
13. Canvel, B., Hiltgen, A., Vaudenay, S., Vuagnoux, M.: Password interception in a SSL/TLS channel. In Boneh, D., ed.: Advances in Cryptology — CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California,

USA, August 17-21, 2003, Proceedings. Volume 2729 of Lecture Notes in Computer Science., Springer-Verlag, Berlin (2003) 583–599
14. Paterson, K.G., Yau, A.: Padding oracle attacks on the ISO CBC mode padding standard. In Okamoto, T., ed.: Topics in Cryptology — CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings. Volume 2964 of Lecture Notes in Computer Science., Springer-Verlag, Berlin (2004) 305–323
15. Vaudenay, S.: Security flaws induced by CBC padding — Applications to SSL, IPSEC, WTLS .... In Knudsen, L., ed.: Advances in Cryptology — EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 – May 2, 2002, Proceedings. Volume 2332 of Lecture Notes in Computer Science., Springer-Verlag, Berlin (2002) 534–545
16. Yau, A.K.L., Paterson, K.G., Mitchell, C.J.: Padding oracle attacks on CBC-mode encryption with secret and random IVs. In: Fast Software Encryption, 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Papers. Lecture Notes in Computer Science, Springer-Verlag, Berlin (2005) to appear
17. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: Proceedings of the 38th IEEE symposium on Foundations of Computer Science, IEEE (1997) 394–403
18. Rogaway, P.: Nonce-based symmetric encryption. In Roy, B., Meier, W., eds.: Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers. Volume 3017 of Lecture Notes in Computer Science., Springer-Verlag, Berlin (2004) 348–359
19. Bellare, M., Kohno, T., Namprempre, C.: Breaking and provably repairing the SSH authenticated encryption scheme: A case study of the encode-then-encrypt-and-MAC paradigm. ACM Transactions on Information and System Security **7** (2004) 206–241
20. International Organization for Standardization Genève, Switzerland: ISO/IEC 9797–1, Information technology — Security techniques — Message Authentication Codes (MACs) — Part 1: Mechanisms using a block cipher. (1999)
21. Barkan, E., Biham, E., Keller, N.: Instant ciphertext-only cryptanalysis of GSM encrypted communications. In Boneh, D., ed.: Advances in Cryptology — CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings. Volume 2729 of Lecture Notes in Computer Science., Springer-Verlag, Berlin (2003) 600–616
22. Ekdahl, P., Johansson, T.: A new version of the stream cipher SNOW. In Nyberg, K., Heys, H., eds.: Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002, St. John's, Newfoundland, Canada, August 15-16, 2002, Revised Papers. Volume 2595 of Lecture Notes in Computer Science., Springer-Verlag, Berlin (2003) 47–61
23. Watanabe, D., Furuya, S., Yoshida, H., Takaragi, K., Preneel, B.: A new keystream generator MUGI. In Daemen, J., Rijmen, V., eds.: Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002, Revised Papers. Volume 2365 of Lecture Notes in Computer Science., Springer-Verlag, Berlin (2002) 179–194
24. International Organization for Standardization Genève, Switzerland: ISO/IEC WD 19772: 2004, Information technology — Security techniques — Authenticated encryption mechanisms. (2004)

25. Rivest, R.L.: All-or-nothing encryption and the package transform. In Biham, E., ed.: Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997, Proceedings. Volume 1267 of Lecture Notes in Computer Science., Springer-Verlag, Berlin (1997) 210–218
26. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable block ciphers. In Yung, M., ed.: Advances in Cryptology — CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings. Volume 2442 of Lecture Notes in Computer Science., Springer-Verlag, Berlin (2002) 31–46