Error oracle attacks and CBC encryption

Chris Mitchell
ISG, RHUL
http://www.isg.rhul.ac.uk/~cjm

Agenda

1. Introduction
2. CBC mode
3. Error oracles
4. Example 1
5. Example 2
6. Example 3
7. Stream ciphers
8. Conclusions

Block ciphers

- One of the fundamental cryptographic primitives is the block cipher.
- When used for encryption, a block cipher takes as input a block of n bits of plaintext P and outputs a block of n bits of ciphertext C. [We call this an n-bit block cipher].
- Operation is controlled by a secret key K; we write $C=e_K(P)$ and $P=d_K(C)$ where d is the decryption function.
- For each key, the encryption function implements a permutation on the set of all n-bit blocks.

Examples of block ciphers

- Some well-known examples of block ciphers:
  - DES: the Data Encryption Standard was first published in the US in the late 1970s, and rapidly became a *de facto* international standard;
  - DES suffers from a relatively short secret key (56 bits), and advances in technology have meant it is unacceptably weak. Triple DES (three iterations of DES using two or three DES keys) is a widely deployed fix to this problem.
  - AES: the Advanced Encryption Standard, is a much more recent design with a 128-bit key, designed as a replacement for DES.

Modes of operation

- Using a block cipher in the naïve way, i.e. dividing the data to be encrypted into blocks, and encrypting each block separately, is not a good idea.
- This is because if two blocks in the plaintext are the same (often likely) then the two ciphertext blocks will be the same.
- That is, the ciphertext will 'leak' information about the plaintext.
- Hence more complex ways of using a block cipher have been devised – called modes of operation.

What is CBC mode?

- CBC (Cipher Block Chaining) mode is a widely used technique for encrypting data using a block cipher (i.e. it is a *mode of operation*).
- It is purely a confidentiality technique – it does not provide any integrity protection for data.
- This is inevitable in that it does not add any redundancy – i.e. *n* bits of plaintext encrypt to *n* bits of ciphertext, so all ciphertexts are 'valid'.

Confidentiality and integrity

- In many cases it is necessary to provide both confidentiality and integrity.
- With symmetric crypto, this is typically achieved by encrypting (e.g. using CBC mode) and computing a MAC (Message Authentication Code).
- Recent cryptanalytic results suggest that these need to be combined with care!

Need for padding

- To use CBC mode, it is necessary for the data that is to be encrypted to be a multiple of *n* bits long (where *n* is the block cipher block length).
- This means that data often needs to be padded prior to encryption.
- Means must be provided for receiver of ciphertext to know which bits of final recovered plaintext are padding.

Padding oracles

- Recipient of ciphertext must process final block to recover and remove padding.
- Depending on padding method, some recovered plaintexts may be ‘invalid’.
- In such a case the decrypter will typically generate an error message, e.g. to request a retransmission.
- This is an example of a *padding oracle*, i.e. an entity which will indicate whether or not a ciphertext yields valid padding when decrypted.

Padding oracle attacks

- Suppose a cryptanalyst can modify/insert messages into a communications channel.
- Then a padding oracle can be used to learn information about the plaintext by repeatedly sending modified versions of the ciphertext to the oracle and seeing what the result is.
- This has been shown to work against real implementations of well-known protocols.

Solutions

- One solution is to try to limit the use of error messages – this is difficult to implement.
- Another widely advocated solution is to use only padding methods for which all possible deciphered messages are valid.
- Most satisfactory solution is to always use an integrity check, and to only decrypt a message if the integrity check passes.

Need for encryption only

- Unfortunately, the final solution is not always practical.
- There are applications where encryption-only is required (these should be minimised).
- Examples include:
  - encrypted voice (telephony) – typically retransmission is not an option because of latency;
  - bulk data transfer (e.g. data trunks) – again retransmission not an option.

Agenda

1. Introduction
2. CBC mode
3. Error oracles
4. Example 1
5. Example 2
6. Example 3
7. Stream ciphers
8. Conclusions

Cipher Block Chaining (CBC) Mode

- Plaintext must be a series of n-bit blocks:

$$P_1,\ P_2,\ \ldots,\ P_q.$$

- Then: $C_1 = e_K(P_1 \oplus IV)$

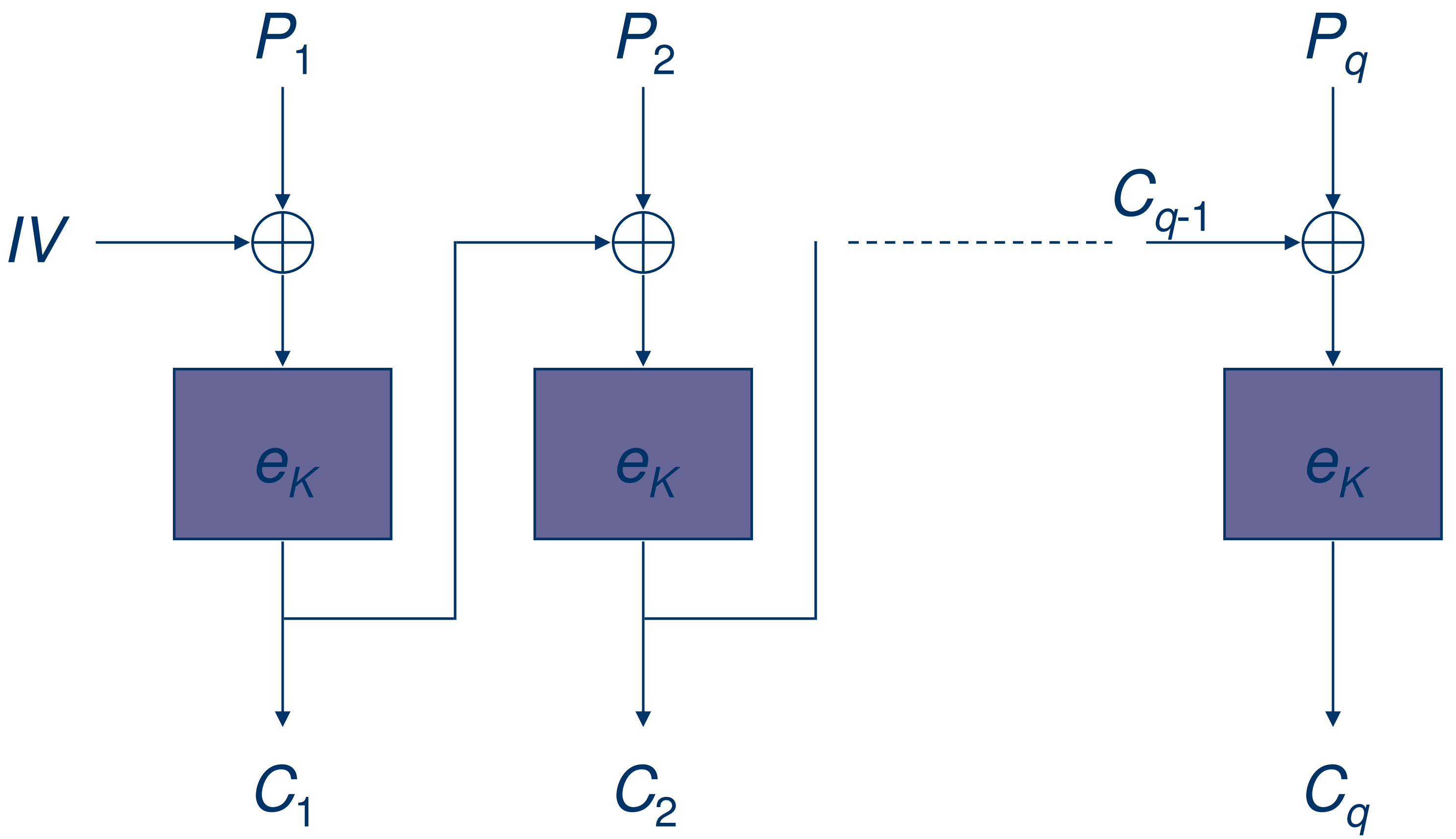
$$C_i = e_K(P_i \oplus C_{i-1}) \quad (i>1)$$

(where $\oplus$ denotes bit-wise exclusive-or), and:

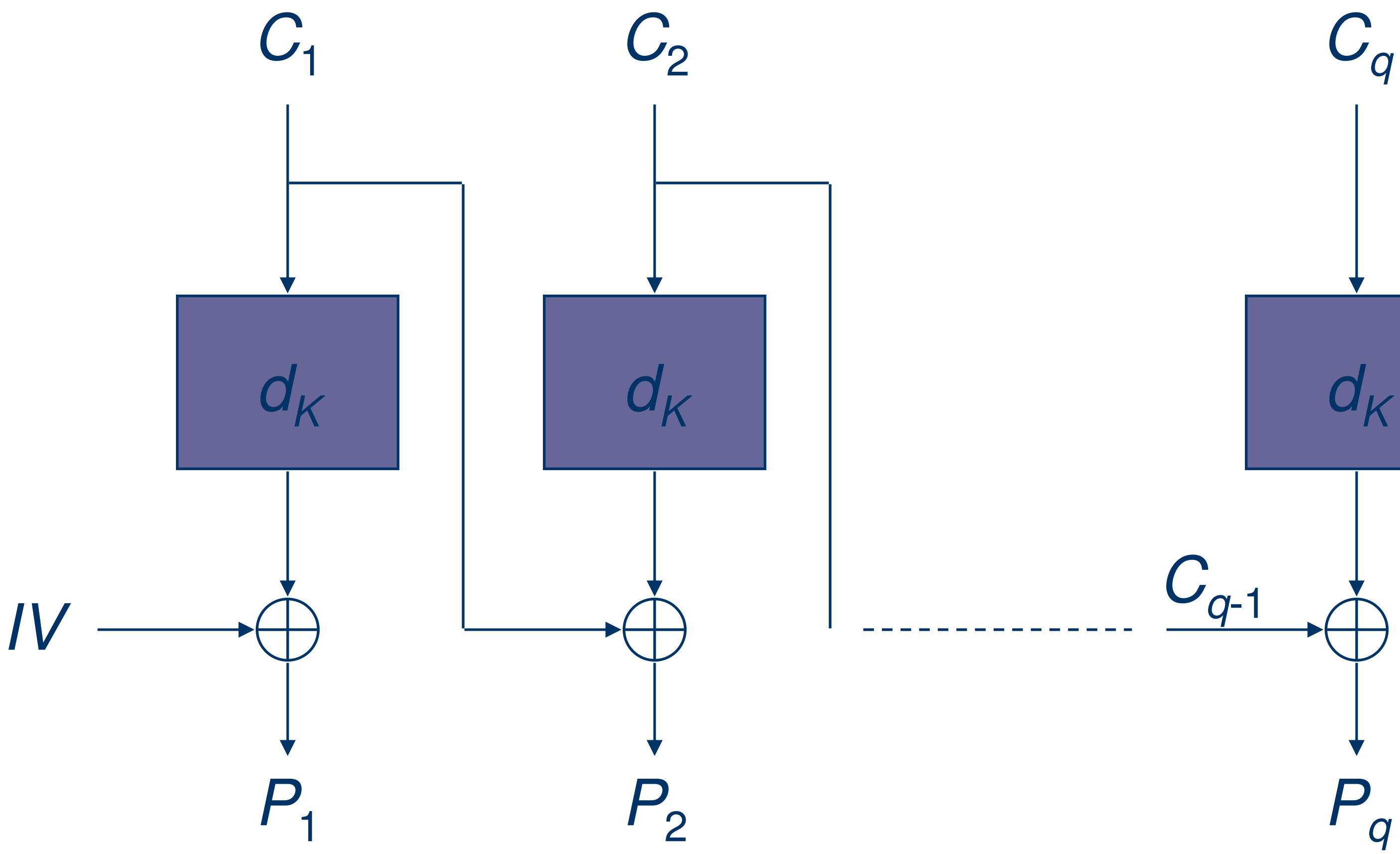
$$P_1 = d_K(C_1) \oplus IV$$

$$P_i = d_K(C_i) \oplus C_{i-1} \quad (i>1).$$

CBC encryption



CBC decryption



CBC mode properties

- If same message encrypted twice using same IV, then the same ciphertext results.
- Two identical plaintext blocks produce different ciphertext blocks.
- Need for padding.
- Error propagation – one bit error in ciphertext means that one block of plaintext is lost, as well as one bit in the next block of plaintext.

An observation

- Suppose $P_1, P_2, \ldots, P_q$ is a (padded) plaintext message which has been encrypted to $C_1, C_2, \ldots, C_q$ using key K and IV S.
- Suppose $X_1, X_2, \ldots, X_{s-1}, C_j, X_{s+1}, \ldots, X_t$ is submitted for decryption, where $s>1$, $j>1$, and decrypted result is $P'_1, P'_2, \ldots, P'_t$.
- Then we have:

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

This observation is key

- This simple observation is the basis of all padding oracle attacks.
- The observation can be use as the basis of two main types of attack designed to learn information about a plaintext message.
- We review these two attack approaches.

Attack type 1

- This attack is designed to learn information about a single 'target' plaintext block P_j.
- Using the previous notation the attacker sets:

$$X_{s-1} = C_{j-1} \oplus Q$$

where Q is a chosen bit pattern.
- By our observation: $P'_s \oplus P_j = Q$, i.e. the attacker can select the difference between P'_s and the target plaintext P_j.
- If the attacker has some means of learning whether or not P'_s generates a formatting error, then he may learn something about P_j.

Attack type 2

- This attack involves learning information about an entire message.
- Suppose C_1, C_2, …, C_q and C^*_1, C^*_2, …, C^*_t are two ciphertext messages (which may be the same) encrypted using the same key.
- The cryptanalyst now submits the message:

$$C^*_1,\ C^*_2,\ \ldots,C^*_{s-1},\ C_j,\ C^*_{s+1},\ \ldots,\ C^*_t$$

- We also suppose that, in this case, the cryptanalyst can force the ‘oracle’ to decrypt this message using the same IV as was used to encrypt C^*_1, C^*_2, …, C^*_t.

Attack type 2 (continued)

- Suppose decrypted result is $P'_1, P'_2, \ldots, P'_t$.
- Then:
  - $P'_i = P^*_i$ for every $i \neq s$ or $s+1$;
  - $P'_s \oplus P^*_s = P^*_s \oplus P_j \oplus C^*_{s-1} \oplus C_{j-1}$;
  - $P'_{s+1} \oplus P^*_{s+1} = C^*_s \oplus C_j$.
- If the attacker has some means of learning whether or not the plaintext generates an error, then this may reveal information about $P^*_s \oplus P_j$ (since everything else is known).

Agenda

1. Introduction
2. CBC mode
3. Error oracles
4. Example 1
5. Example 2
6. Example 3
7. Stream ciphers
8. Conclusions

Padding oracles reviewed

- In a padding oracle attack, an attacker has one or more valid ciphertexts, and can inject modified ciphertexts into the channel.
- The receiver will decrypt each ciphertext and generate an error message if the padding is incorrect.

Error oracles

- In an *error oracle* attack we suppose that, after decryption, the message is passed to a protocol implementation (e.g. an application) which will generate a detectable action (e.g. an error message) if the message format is incorrect.
- In this sense a padding oracle attack is just a special case of an error oracle attack.

Discussion

- Unlike padding oracles, it may not be possible to prevent error oracles.
- Applications are run across encrypted networks, where the application is not encryption-aware and the encryption layer is not application-aware.
- It is inevitable that some applications will react in unexpected ways to ill-formatted messages.
- Hence likelihood of error oracles should be minimised, e.g. by using authenticated encryption whenever possible.

Agenda

1. Introduction
2. CBC mode
3. Error oracles
4. Example 1
5. Example 2
6. Example 3
7. Stream ciphers
8. Conclusions

Assumptions

- Suppose a protocol, running at a higher layer in the protocol hierarchy than the encrypting protocol, provides error protection using a 16-bit CRC.
- I.e. suppose plaintext $P_1, P_2, \ldots, P_q$ corresponding to ciphertext $C_1, C_2, \ldots, C_q$, incorporates a 16-bit CRC.
- Suppose attacker can also find out if error detection fails.

The error oracle query (attack type 2)

- The attacker replaces C_s with C_j for some $s \neq j$.
- If the recovered 'plaintext' is $P'_1, P'_2, \ldots, P'_t$, then:
  - $P'_i = P_i$ for every $i \neq s$ or $s+1$;
  - $P'_s \oplus P_s = P_s \oplus P_j \oplus C_{s-1} \oplus C_{j-1}$;
  - $P'_{s+1} \oplus P_{s+1} = C_s \oplus C_j$.
- Given the original message contained a valid CRC, then the corrupted message will contain a valid CRC if and only if the exor of the original and corrupted messages contains a valid CRC (by linearity).

Results

- The exor of the original and corrupted plaintexts will be zero in all but two blocks, and the only unknown for these two blocks is the value of $P_s \oplus P_j$.
- The probability the CRC will be correct is 2^{-16}, but in that case the attacker will instantly know 16 bits of information about the message.
- If an 8-bit CRC is used, then information can be obtained more rapidly.

Agenda

1. Introduction
2. CBC mode
3. Error oracles
4. Example 1
5. Example 2
6. Example 3
7. Stream ciphers
8. Conclusions

A message structure attack (type 1)

- Suppose the target plaintext message contains a fixed byte in a known position.
- Suppose the fixed byte is the *k*th byte of block P_s, for some $s > 1$.
- Many protocols contain fixed bytes (e.g. set to zero) for future-proofing – perhaps containing the version number of the protocol.

The error oracle query

- The attacker constructs 256 queries, one for each value of t ($0 \le t \le 255$).
- The attacker replaces C_s with C_j for some $j \neq s$, and replaces C_{s-1} with $C_{j-1} \oplus Q_t$, where Q_t has zeros everywhere except in the kth byte, which contains the binary representation of t.
- Precisely one of these (Q_u say) will yield a plaintext with the correct value for the kth byte of the sth plaintext block.

Results

- By the key observation, the recovered plaintext block P'_s will equal:

$$P_j \oplus Q_u$$

- That is, for the value of t (i.e. u) that does not yield an error, the attacker knows that the kth byte of $P_j \oplus Q_u$ will equal the correct fixed byte.
- This immediately gives a byte of p/text block P_j.
- Repeat for every plaintext block (except P_1).

Agenda

1. Introduction
2. CBC mode
3. Error oracles
4. Example 1
5. Example 2
6. <u>Example 3</u>
7. Stream ciphers
8. Conclusions

A content-based padding oracle attack

- We now describe a padding oracle attack which works against padding methods which are resistant to ‘normal’ padding oracle attacks.
- We need to suppose that the message sent is of fixed length, and that an error message will be generated if a message is received of the wrong length.

Assumptions

- We suppose that the padding method in use involves adding a single one to the end of the data followed by the smallest number of zeros (at most n-1) necessary to create a whole number of n-bit blocks.
- This padding method is uniquely unpaddable, and resists known padding oracle attacks (almost every possible string of bits corresponds to a padded message).

The error oracle query

- Suppose C_1, C_2, …, C_q is a valid ciphertext message for which the last d bits of P_q are 100…0 (the fixed message length is $qn-d$).
- The attacker makes 2^d messages variants ($0 \le t \le 2^d-1$) by modifying the last two blocks to:

$$C_{j-1} \oplus Q_t,\ C_j$$

where Q_t contains $n-d$ zeros followed by the binary representation of t.

- One will not return a message length error – say Q_u.

Results

- By the key observation, the recovered plaintext block P'_q will equal:

$$P_j \oplus Q_u$$

- That is, for the value of t (i.e. t=u) that does not yield an error, the attacker knows that the final d bits of $P_j \oplus Q_u$ will equal 100…0.
- This immediately gives d bits of p/text block P_j.
- Repeat for every plaintext block (except P_1).

Agenda

1. Introduction
2. CBC mode
3. Error oracles
4. Example 1
5. Example 2
6. Example 3
7. <u>Stream ciphers</u>
8. Conclusions

CBC mode and stream ciphers

- It would thus appear that CBC mode is dangerously prone to error oracle attacks, regardless of the padding method used.
- One other widely used method of encryption is the *stream cipher*.
- In a stream cipher, the data is encrypted by bit-wise exoring it with a pseudorandom *keystream* sequence (generated as a function of a secret key).

Stream ciphers and error oracles

- Stream ciphers do not suffer in the same way (they also do not require padding).
- There are examples of error oracle attacks on stream ciphers, but they seem harder to construct.
- Suppose two consecutive bits of a plaintext message are always equal to one of 00, 01, and 10 (and that 11 will cause a detectable behaviour by the recipient).
- If the second of the two corresponding ciphertext bits is changed then error/no error means that the previous bit is 1/0.

Agenda

1. Introduction
2. CBC mode
3. Error oracles
4. Example 1
5. Example 2
6. Example 3
7. Stream ciphers
8. Conclusions

Use authenticated encryption

- The simplest and best solution to all these attacks is to use authenticated encryption (AE).
- This either means use the 'encrypt-then-MAC' paradigm, or use one of the AE block cipher modes recently developed (OCB versions 1 and 2, EAX, CCM, …).
- Indeed, an international standard for AE schemes, ISO/IEC 19772, is being developed.

Use a stream cipher

- If unauthenticated encryption is really necessary, then don't use CBC mode!
- Probably the best choice is a stream cipher.
- This either means using a bespoke keystream generator (e.g. SNOW 2.0 or MUGI) or a block cipher in an appropriate mode, e.g. OFB or CTR mode.

Acknowledgements

- Must thank Kenny Paterson for many helpful comments.
- A shorter version of this talk will be presented at ISC 2005 (Singapore, September 2005).