Serge Vaudenay in 2002 introduced padding oracle attacks, showcasing a significant vulnerability in CBC mode encryption [1]. This review traces the development of padding oracle attacks, underlining the constant interaction between cryptographic research and standardization efforts.

**Vaudenay's Original Work**

Vaudenay's seminal paper [1] exposed the vulnerability in CBC mode encryption when a padding oracle is present. A padding oracle exists when an attacker can distinguish between correct and incorrect padding, allowing for iterative decryption of ciphertexts without the encryption key. Vaudenay demonstrated that, with access to a padding oracle, an attacker could efficiently decrypt data by exploiting the error messages returned from the oracle.

The attack, can be summarized as follows:

**Algorithm (Simplified Description):**

1. **Input**: Ciphertext *C* divided into blocks *C*1​,*C*2​,...,*Cn*​.
2. **Output**: Plaintext *P*.
3. For each block *Ci*​ from *Cn*​ to *C*1​:

a. Modify *Ci*−1​ in various ways and observe the oracle's response to the padding validity of *Ci*​.

b. Deduce the value of *Pi*​ based on the oracle's feedback.

This attack fundamentally challenged the security assumptions of CBC mode encryption, particularly in protocols like SSL/TLS, where error messages could inadvertently serve as padding oracles.

The attack has a complexity of *O*(*N,b,W*), where *N* is the number of blocks, *b* is the block size, and *W* is the set of possible word values (typically 256256 for a byte). This makes the attack surprisingly efficient.

**Expansion and Adaptation**

**A. Padding Oracle Attacks on the ISO CBC Mode Encryption Standard**

Paterson and Yau's 2004 research [2] extended Vaudenay's original attack to the International Organization for Standardization (ISO) CBC-mode encryption standard, scrutinizing the security of specific padding methods recommended by ISO/IEC 9797-1 [4] and ISO/IEC 10118-1[5]. They uncovered vulnerabilities in several of these padding methods, demonstrating that the straightforward application of padding oracle attacks could lead to efficient plaintext recovery.

The first two padding methods mentioned in [4] are very primitive and very easily exploitable compared to the third padding method which we can restate as:

"The data string *D* to be input to the [...] algorithm shall be right-padded with as few (possibly none) '0' bits as necessary to obtain a data string whose length (in bits) is a positive integer multiple of n. The resulting string shall then be left-padded with a block *L*. The block *L* consists of the binary representation of the length (in bits) of the unpadded data string *D*, left-padded with as few (possibly none) '0' bits as necessary to obtain an *n*-bit block. The right-most bit of the block L corresponds to the least significant bit of the binary representation of ."



Figure-1: ISO/IEC 9797-1 padding method 3

The attack in [4, Section 3.4] decrypts, one block at a time, arbitrary ciphertexts C1|| C2|| …. || Cq that are padded using the above method. It requires *n* + O() oracle calls per block. The attack has 2 phases. In the first phase the is determined. An algorithm “9797-1-m3-get-LD-general” is used which can find when a q-block valid ciphertext is given as input by manipulating the padding bits. For successful operation of this algorithm the re-use of old IVs is required. In the general form of this algorithm the number of blocks (q) considered is equal or more than three.

Algorithm : 9797-1-m3-get-LD-general

Input: IV||C1||C2||...||Cq

Output:

Ensure: q ≥ 3

*C := IV||C1||C2||...||Cq*

*l := 0*

*u := n − 1*

*repeat*

*h := ⌊(l + u)/2⌋*

*Cq-1,h := Cq-1,h ⊕ 1*

*if oracle(C) = VALID then*

*l := h*

*else if oracle(C) = INVALID then*

*u := h−1*

*end if*

*Cq-1,h := Cq-1,h ⊕ 1*

until l = u

return := (q − 1)n + l + 1

There is a special case of this algorithm which applies to two-block ciphertexts only (*q*=2).

The second phase of the attack is the actual decryption. The decryption algorithm “797-1-m3-decrypt” returns all the bits of a plaintext block except the last one, and in doing so it makes repeated updates to the IV.

Algorithm: 9797-1-m3-decrypt

Input: , IV, C1, Ck

Output: Pk,1, Pk,2, ... , Pk,n−1, the rightmost n − 1 bits of Pk

R := a random *n-*bit block

for j := n − 1 to 1 do

*IV' := IV ⊕ ⊕ (n + j)2*

*b := Ω(C')*

*C' := IV'||C1||R||Ck*

*Pk,j := b ⊕ Rj ⊕ Ck−1,j*

*R := R ⊕ 0...0 b 0...0*

end for

return *Pk,1, Pk,2, ... , Pk,n−1*

There is another algorithm “9797-1-m3-decrypt-last-bit” written to recover the last bit, but it requires the successful run of the 797-1-m3-decrypt algorithm.

Algorithm 9797-1-m3-decrypt-last-bit

Input: Ck−1, Ck, Pk,1, Pk,2, ... , Pk,n

Output: Pk,0, the leftmost bit of Pk

*R := a random n-bit block*

*IV' := Ck−1 ⊕ 0Pk,1Pk,2...Pk,n ⊕ (n)2*

*C' := IV'||Ck||R*

*Pk,0 := Ω(C')*

return Pk,0

In this attack optimal usage of the padding oracle is ensured which helps in decrypting many ciphertexts for the same key K. The work also highlighted the risks of adopting padding methods from one standard to another without thorough security reassessment. The authors also proposed that the integrity of encrypted data could be improved by employing authenticated encryption, thereby mitigating the effectiveness of padding oracle attacks.

**Attacking the padding methods of ISO/IEC 10118-1**

In this work padding method 3 is considered for analysis and attack purpose as method 1 and 2 are very easy to exploit and has simple vulnerabilities. The original statements of padding method 3 from [5] is restated here and instead of using for block size, *n* is used.

"This padding method requires the selection of a parameter *r* (where *r ≤ n*), e.g*. r* = 64, and a method of encoding the bit length of the data D, i.e. as a bit string of length *r*. The choice for *r* will limit the length of D, in that < .

"The data D [...] is padded using the following procedure.

1. D is concatenated with a single ‘1’ bit.
2. The result of the previous step is concatenated with between zero and *n - 1* ‘0’ bits, such that the length of the resultant string is congruent to *n - r* modulo *n*. The result will be a bit string whose length will be *r* bits short of an integer multiple of n bits (in the case *r = n*, the result will be a bit string whose length is an exact multiple of n bits).
3. Append an *r*-bit encoding of using the selected encoding method, yielding the padded version of D."

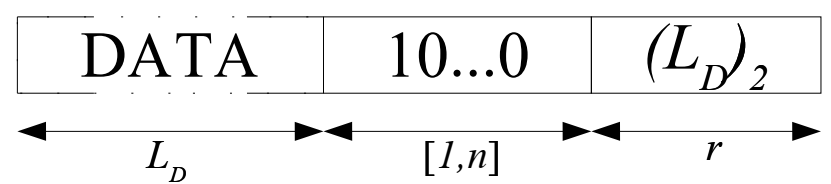


Figure 2: ISO/IEC 10118-1 padding method 3

Using this padding method, the padding bits for data string *D* are appended in one of two ways:

Same-block (mod *n* ≤ (*n*−*r*−1)) The last block has enough space after the last plaintext bit to contain at least a single ‘1’ bit and the *r* bits of *L*, the length block that holds . The number of padded bits is between *r*+1 and *n*−1.

Cross-block (mod *n* ≥ (*n*−*r*)) The last block does not have enough space to contain a ‘1’ bit and the *r* bits of *L*. The number of padded bits is between *n* and *n*+*r* and the padding extends over two blocks. Note that this will always be the case when *r*=*n*.

In their research [4], the authors discussed two interconnected attacks targeting this specific padding method. One attack generates a legitimate ciphertext where the target ciphertext block is positioned as the final block. Meanwhile, the other attack is capable of decrypting the last block of any given ciphertext.

**Attack 1: Directed IV Search**

This attack canrecover any plaintext block from *Ck*​ using a carefully selected Initialization Vector (IV). It is effective when the padding parameter *r* ≤ *n*−1. For *r* = *n*, a modified approach is needed.

**Steps:**

1. Start with a random IV and modify its rightmost *r* bits in all possible combinations. Submit these modified ciphertexts to the oracle.
2. For each modified IV, construct a ciphertext *C*=*IV*∣∣*Ck*​ and submit it to the padding oracle. If the oracle returns "VALID", this IV and *Ck*​ combination is correctly padded.
3. Once a valid padding is found, use the same approach as in Vaudenay’s attack to decrypt *Ck*​, adjusting the IV as needed to make *C*=*IV*∣∣*Ck*​ appear as a single-block message with correct padding.
4. The expected number of oracle calls is for *r* <*n*, ensuring at least one "VALID" response. For *r* = *n*, exploring all settings guarantee a valid plaintext block recovery.

**Attack 2: Attacking the Last Block(s)**

This attack efficiently recover plaintext bits from the last block(s) that contain both data and padding bits.

**Steps:**

1. Determining : Use a binary search method to find the exact length of the unpadded data by flipping bits from the right towards the left in the last ciphertext block and observing the oracle's responses. This step identifies the boundary between data and padding.
2. Decrypting the Data Bits:
   * For **Same-Block Padding**, directly apply the decryption algorithm to reveal the plaintext bits in the last block by adjusting the IV to decrement the length field bit by bit.
   * For **Cross-Block Padding**, where padding extends over the last two blocks, a variant of the directed IV search is used with knowledge of some padding bits to recover the penultimate plaintext block more efficiently.
3. Determining requires oracle calls. Decrypting the data bits in the last block requires an additional oracle call for each bit. The total number of required oracle calls is *t*+ for same-block padding, where *t* is the number of data bits in the last block.

These attacks exploit the padding oracle's feedback mechanism to systematically deduce the plaintext without direct access to the encryption key, demonstrating the critical need for secure padding methods and the careful design of cryptographic protocols to mitigate such side-channel vulnerabilities.

**B. Padding Oracle Attacks with Secret and Random IVs**

Yau et al introduces new padding oracle attacks within the context of the ISO CBC-mode encryption standard, specifically when Initialization Vectors (IVs) are kept secret and random. This research addresses and circumvents the protective measures recommended by a revised draft of the ISO standard, which aimed to mitigate previous padding oracle vulnerabilities by suggesting the use of secret, random IVs. As like the previous research this work focuses on Padding Method 3 of ISO/IEC 9797-1. It Leverages auxiliary ciphertexts with distinct lengths and manipulates them to ascertain the correct padding of a target ciphertext block. Previous attacks (including Vaudenay's and Paterson and Yau's earlier work) could directly manipulate or had knowledge of the IV, significantly aiding in the decryption process. This new attack adapts to the constraints of secret and random IVs by using external ciphertext characteristics to infer padding information, thus sidestepping the need to know or manipulate the IV directly.

For Padding Method 3 of ISO/IEC 10118-1 the designed attack also operates under the assumption of secret and random IVs. The technique is to construct valid ciphertexts with the target block as the final block, which can then be decrypted to recover plaintext bits.

This research underscores a critical insight: the mere adoption of secret and random IVs is insufficient to fully mitigate the risk of padding oracle attacks. It illustrates that the security of CBC-mode encryption, even when adhering to ISO standard recommendations, can be compromised unless padding methods are also robust against such side-channel attacks. Consequently, this work contributes to the ongoing discussion about the need for comprehensive security strategies that encompass both the encryption mode and the padding techniques used, reinforcing the call for the careful selection of padding methods and the implementation of strong integrity checks where feasible.

**References**

[1] S. Vaudenay, "Security Flaws Induced by CBC Padding - Applications to SSL, IPSEC, WTLS...," in *Advances in Cryptology — EUROCRYPT 2002*, Springer, Berlin, Heidelberg, 2002, pp. 534-545.

[2] K. G. Paterson and A. Yau, "Padding Oracle Attacks on the ISO CBC Mode Encryption Standard," in *Topics in Cryptology — CT-RSA 2004*, Springer, Berlin, Heidelberg, 2004, pp. 305-323.

[3] A. K. L. Yau, K. G. Paterson, and C. J. Mitchell, "Padding Oracle Attacks on CBC-mode Encryption with Secret and Random IVs," in Proceedings, 2005

[4] ISO/IEC 9797-1: Information technology | Security tehniques | Message Authentication Codes (MACs) - Part 1: Mechanisms using a block cipher. 1999.

[5] ISO/IEC FDIS 10118-1: Information technology - Security techniques - Hash-functions - Part 1: General (Final Draft). 2000