# On the security of CBC Mode in SSL3.0 and TLS1.0*

Takashi kurokawa† , Ryo Nojima, and Shiho Moriai
National Institute of Information and Communications Technology, Koganei, Tokyo, Japan
{blackriver, ryo-no, shiho.moriai}@nict.go.jp

**Abstract**

Currently, SSL (Secure Socket Layer) and TLS (Transport Layer Security) are two of the most widely used security protocols on the Internet and TLS1.0 is one of the most supported protocol versions through SSL/TLS. To protect the application data in SSL3.0/TLS1.0, two bulk data encryption algorithms are selected by the ciphesuites of them: the stream cipher encryption or the block cipher encryption in combination with the cipher block chaining (CBC) mode of operation. For these several years, they have been criticized to be insecure when used in the real world. For example, the BEAST attack against TLS1.0 and the POODLE attack against SSL3.0 had a significant impact on the internet security not least because their techniques are clever and their computational costs are low. In this paper, we survey their attacks and prove theoretically that the patched CBC mode in TLS1.0 satisfies *indistinguishability*, which implies that it is secure against BEAST type of attack.

**Keywords**: SSL3.0, TLS1.0, CBC Mode, The BEAST attack, Security

## 1 Introduction

SSL/TLS is one of the most widely used cryptographic protocols on the Internet. *SSL3.0* [16] was released by Netscape Communications in 1996 and then *TLS1.0* [9] was released by Internet Engineering Task Force (IETF) in 1999. Updated versions *TLS1.1* [10] and *TLS1.2* [11] were released in 2006 and 2008, respectively.

Currently, SSL/TLS has been deployed to securely perform almost all the popular network services, for example, online shopping and online banking. At same time, many cryptographic attacks against SSL3.0 and TLS1.0 have been found, e.g., CRIME [30], Lucky Thirteen [2], BEAST [13], POODLE [26] and RC4 biases attacks [1, 18].[1]

In SSL 3.0/TLS 1.0, many cryptographic primitives have been employed, e.g., RSA [32], DH(E)[2] [12], AES [27], RC4 [34], CBC mode [14], and HMAC [28]. Among them, we are going to focus on the *CBC mode* in SSL3.0/TLS1.0, which can cause security issues in SSL/TLS.

According to the SSL Pulse data [36], TLS1.0 is currently the most widely used protocol version through SSL/TLS[3], and the CBC mode is included in many ciphersuites of them[4]. Although a software patch was released for the CBC mode, there still remains a problem. Namely, it is not clarified whether or not the patched CBC mode is secure against a BEAST type of attack, or how secure it is.

*Journal of Internet Services and Information Security (JISIS)*, volume: 6, number: 1 (February 2016), pp. 2-19

*This paper is a full version of the work originally presented at the 2015 Asian Conference on Availability, Reliability and Security (AsiaARES'15), Daejeon, Republic of Korea, September 2015 [20].

†Corresponding author: Security Fundamentals Laboratory, Network Security Research Institute, National Institute of Information and Communications Technology (NICT), 4-2-1, Nukui-Kitamachi, Koganei, Tokyo 184-8795, Japan, Tel: +81-42-327-5803

[1]For the overview of these attacks, see [33].

[2]DHE denotes ephemeral Diffie-Hellman key exchange.

[3]As of December 2015, 98.8% of the sites surveyed by the SSL Pulse support TLS1.0.

[4]As of December 2015, 0.3% of the sites surveyed by the SSL Pulse support only RC4 ciphersuites.

Although we have showed that the patched CBC mode satisfies indistinguishability in our previous paper [20], we need to clarify a relation between indistinguishability and security against a BEAST type of attack. In this extended paper, we will demonstrate these connections in addition to related topics for the attacks. The rest of the paper is organized as follows. In Section 2 we give a brief overview of SSL3.0 and TLS1.0 and the attacks against them. In Section 3 we describe the preliminaries. In Section 4, we describe the original CBC mode and the patched CBC mode and discuss the connection between indistinguishability of the patched CBC mode and BEAST type of attack. In Section 5, we present the proof of indistinguishability. In Section 6, we conclude the paper.

# 2 SSL3.0 and TLS1.0

## 2.1 Overview of the Record Layer

SSL3.0 and TLS1.0 consist mainly of the *Record Protocol*, the *Handshaking Protocol*, the *Change Cipher Spec Protocol* and the *Alert Protocol*. We omit explanations of other protocols than the record protocol. By the record protocol, the data passed by higher layers is fragmented to information blocks. One of the specified information blocks is the *SSLCiphertext* or the *TLSCiphertext* as shown below (See Listing 1, 2). In case that the block cipher encryption is chosen by the *CipherSuite* in the SSL/TLS, the *GenericBlockCipher* is selected as the *fragment*.

Listing 1: SSLCiphertext structure in SSL3.0

```
struct {
    ContentType type;
    ProtocolVersion verions;
    uint16 length;
    select (CipherSpec.cipher_type) {
        case stream: GenericStreamCipher;
        case block: GenericBlockCipher;
    } fragment;
} SSLCiphertext;

block-ciphered struct {
    opaque Content[SSLCompressed.length];
    opaque MAC[CipherSpec.hash_size];
    uint8 padding[GenericBlockCipher.padding_length];
    uint8 padding_length;
} GenericBlockCipher;
```

Listing 2: TLSCiphertext structure in TLS1.0

```
struct {
    ContentType type;
    ProtocolVersion verions;
     uint16 length;
     select (CipherSpec.cipher_type) {
         case stream: GenericStreamCipher;
         case block: GenericBlockCipher;
    } fragment;
} TLSCiphertext;

block-ciphered struct {
    opaque Content[TLSCompressed.length];
    opaque MAC[CipherSpec.hash_size];
    uint8 padding[GenericBlockCipher.padding_length];
    uint8 padding_length;
} GenericBlockCipher;
```

The *GenericBlockCipher* is encrypted by the selected block cipher in combination with the CBC mode of operation. In order to force the total length of this block structure to be an integral multiple of the block size of the chosen block cipher, the *padding* is appended just after the *MAC*.

In SSL3.0, each byte of the *padding* is indefinite and the *padding_length* is less than the block cipher's block size or zero. In TLS1.0, each byte of the *padding* is filled with the *padding_length* value and the padding length is up to 255 bytes long.

The *MAC* is generated by the following computation (See Listing 3, 4). Note that the *seq_num* denotes the sequence number of the record.

Listing 3: MAC in SSL3.0

```
hash(MAC_write_secret + pad_2 +
     hash(MAC_write_secret + pad_1 + seq_num +
          SSLCompressed.type + SSLCompressed.length +
          SSLCompressed.fragment));
where "+" denotes concatenation.
```

Listing 4: MAC in TLS1.0

```
HMAC_hash(MAC_write_secret, seq_num + TLSCompressed.type +
              TLSCompressed.version + TLSCompressed.length +
              TLSCompressed.fragment));
where "+" denotes concatenation.
```

## 2.2 The CBC Mode in SSL3.0 and TLS1.0

In the SSL/TLS, a plaintext is "tagged" before the encryption and the tag is computed as the *MAC*, as described above. In other words, before encrypting a plaintext $M$, the tag $t$ is firstly generated and secondly appended to $M$, then the message

$$M' = M \| t$$

is thirdly encrypted using the CBC mode (See Figure 1). Then, the ciphertext of the (tagged) message

$$M' = (M'[0], M'[1], \ldots, M'[m-1])$$

is represented as

$$\mathscr{F}_K(\mathtt{IV} \oplus M'[0]), \ldots, \mathscr{F}_K(C[m-2] \oplus M'[m-1] \| \mathtt{padding} \| \mathtt{padding\_length}), \tag{1}$$

where $\mathscr{F}_K : \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a secure block cipher which can be modeled as the pseudorandom permutation (See Section 3.1), $\lambda$ is the block length, `IV` is an initial vector, `padding` is the padding of the GenericBlockCipher structure, `padding_length` is the length of `padding`,

$$C[i] = \begin{cases} \mathscr{F}_K(\mathtt{IV} \oplus M'[0]) & \text{for } i = 0, \\ \mathscr{F}_K(C[i-1] \oplus M'[i]) & \text{for } 1 \le i \le m-2, \\ \mathscr{F}_K(C[m-2] \oplus M'[m-1] \| \mathtt{padding} \| \mathtt{padding\_length}) & \text{for } i = m-1. \end{cases}$$

The `IV` for the first record is generated from the *SecurityParameters* in the the SSL/TLS and an `IV` of the next record is set from a ciphertext block of a previous record.

The CBC mode in the SSL/TLS has two potential weaknesses: one is in the padding scheme and the other is in the handling of the initial vector `IV` [25].

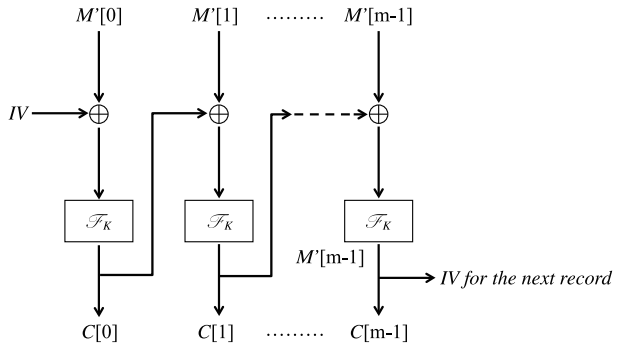


Figure 1: The CBC mode in SSL 3.0 and TLS 1.0

*Padding*: In the encryption of the form Eq.(1), which follows the Mac-then-Enc paradigm [17], the message authentication code is not applied to the padding. In other words, the padding is appended after the generation of the tag. When catching an error in this form, we can consider two cases: the error of the padding scheme and that of the message authentication code. If these two cases can be distinguished by an adversary, an attack which is known as the *padding oracle attack* [37] may work. For example, a timing analysis [8] enables the adversary to distinguish these two cases. There is also a possibility that other side channel information can be exploited to attack the CBC mode. In fact, the Möller et al. [26] found a practical attack against the CBC mode in SSL3.0, named the POODLE attack (See Section 2.3.2). Because TLS employs a different padding scheme than that of SSL, this attack cannot be applied directly to the CBC mode in TLS. However, some implementations of TLS was affected by the POODLE attack [22].

*Choice of* `IV`: In SSL 3.0 and TLS 1.0, the initial vector `IV` after the first record is chosen from the last block of the ciphertext, therefore the adversary who can eavesdrop the ciphertexts knows the `IV` before the next plaintext is encrypted [31]. Since this means that `IV` is predictable from the adversary's viewpoint, the CBC mode in SSL 3.0 and TLS 1.0 does not satisfy indistinguishability. This fact does not immediately imply that the adversary can recover the whole plaintext, and moreover it would be expected that the time complexity of the recovering the plaintext would be $O(2^{\lambda})$ for one block of ciphertexts. However, Duong and Rizzo demonstrated the BEAST attack [13] whose time complexity is $O(\lambda)$.

In TLS1.1 [10] and TLS1.2 [11], the subsequent `IV` is securely sent from one to the other by the following structure (See Listing 5).

Listing 5: GenericBlockCipher structure in TLS1.1 and TLS1.2

```
block-ciphered struct {
      opaque IV[CipherSpec.block_length];
      opaque content[TLSCompressed.length];
      opaque MAC[CipherSpec.hash_size];
      uint8 padding[GenericBlockCipher.padding_length];
      uint8 padding_length;
} GenericBlockCipher;
```

## 2.3 Attacks against SSL3.0 and TLS1.0

In this section, we give outlines of the BEAST attack in addition to the POODLE attack and RC4 biases attacks by way of comparison.

Some web browsers and web application programming interfaces (APIs) adopt *Same-Origin Policy* (SOP) [6] to restrict interactions between different domains. Therefore, a security flaw in SOP involves

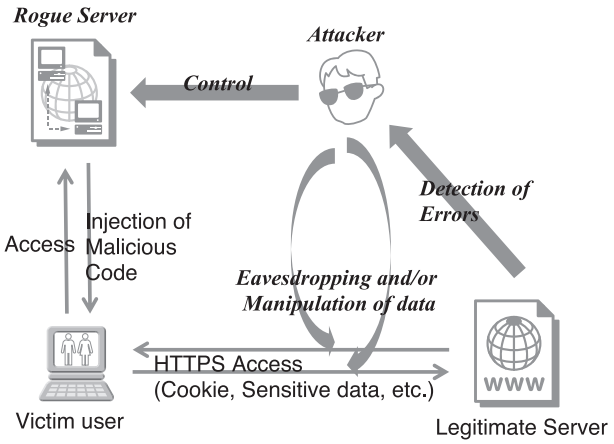


Figure 2: Schematic View of Attack Scenarios

substantial risk of data theft such as secret information typed into legitimate web sites and HTTP cookies [5] by an attacker.

In the attacks cited below, we make an implicit assumption that a target of an attacker knows a vulnerability of SOP in a web browser of a target, etc..

Let us suppose that an attacker can control the following actions of the target (See Figure 2):

S1. To direct a target to a rogue web server under an attacker's control and access to it.

S2. To infect a target with malicious code inserted into a web page on a rogue web server.

S3. To direct a target to a legitimate web server and access to it.

S4. To make a target send his secret information to a legitimate web server.

### 2.3.1 The BEAST Attack

The BEAST attack[5] [13] is the chosen-plaintext attack against SSL3.0 and TLS1.0 and is mainly divided into two phases described below (See Figure 3):

B1. Chosen Boundary Phase:

  (a) An attacker generates a string of a certain length and sends it to a target.

  (b) A target (unknowingly) prepends it to a plaintext to be encrypted and encrypts the resulting text in CBC mode and sends the ciphertext to a server.

  (c) An attacker knows the ciphertext.

B2. Blockwise Phase:

  (a) An attacker XORs a certain block of a plaintext, a certain and the last block of a ciphertext and sends it to a target.

  (b) A target (unknowingly) encrypts the resulting text and sends the ciphertext to a server.

[5]BEAST stands for "Browser Exploit Against SSL/TLS".

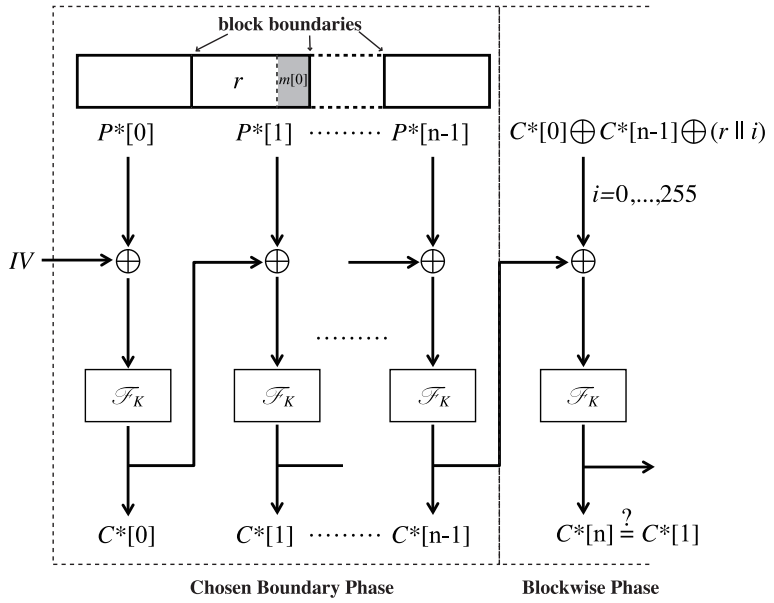


Figure 3: Schematic View of the BEAST Attack

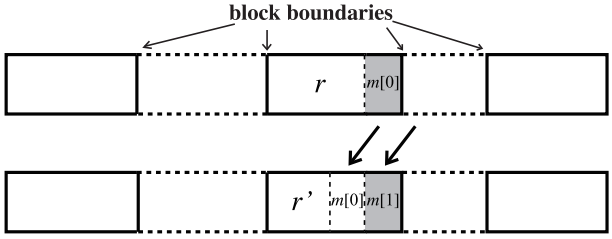


Figure 4: Sequential Alignment in the Chosen Boundary Phase in the BEAST Attack

(c) An attacker repeats the above procedures until a certain condition is satisfied.

We refer to such a attack as BEAST type of attack. Note that, in the first phase, it is crucial to make the first unknown byte put the last byte of a certain (but not first) block of a plaintext and, in the second phase, an attacker can learn the `IV` of the next record as the last block of the last ciphertext though he does not know the first `IV`. Moreover, an attacker can learn the unknown byte $m[0]$ if he continues to encrypt forged blocks $C^*[0] \oplus C^*[n-1] \oplus (r||i)(i = 0, \ldots, 255)$ until the resulting encrypted block $C^*[n]$ equals to a former block $C^*[1]$ (See Figure 3).

In fact, at the second block of the first record, we have

$$\mathscr{F}_K^{-1}(C^*[1]) = C^*[0] \oplus P^*[1] = C^*[0] \oplus (r||m[0]),$$

and at the first block of the second record, thanks to the cancellation of $C^*[n-1]$, we have

$$\mathscr{F}_K^{-1}(C^*[n]) = C^*[n-1] \oplus (C^*[0] \oplus C^*[n-1] \oplus (r||i)) = C^*[0] \oplus (r||i).$$

Then $m[0] = i$ when the equality $C^*[n] = C^*[1]$ holds.

The other unknown bytes of a plaintext can be decrypted if an attacker adjusts a length of a prepended string in the manner as mentioned above (See Figure 4).

To mount the BEAST attack, two underlying conditions are necessary. One is that there exists a vulnerability to bypass Same Origin Policy (SOP) in the browser and the other is the predictability of IV, which is the case of the CBC mode in SSL3.0/TLS1.0. The attack had a huge impact since Duang and Rizzo presented the software bug on SOP in Java. There is a possibility that there are many software bugs other than Java. Hence, browser vendors such as Google, Microsoft and Mozilla released a software patch for the CBC mode in addition to the patch for Java [33].

### 2.3.2 The POODLE Attack

The POODLE attack[6] [26] is the man-in-the-middle attack against CBC mode in SSL3.0, which exploits weakness of validity check of the padding scheme, and is mainly divided into two phases described below (See Figure 5):

P1. Chosen Boundary Phase:

(a) An attacker generates strings and sends it to an target.

(b) A target (unknowingly) inserts them to a plaintext so that only an unknown byte is put at the last byte of a certain block and a boundary between a rightmost byte of the MAC and a leftmost byte of the padding becomes a block boundary.

(c) A target encrypts the resulting text in CBC mode and sends the ciphertext to a server.

P2. Blockwise Phase:

(a) An attacker substitutes a certain block of the ciphetext including the unknown byte for the last block of the ciphertext.

(b) An attacker continues the above phases until a server can decrypt the manipulated ciphertext without errors.

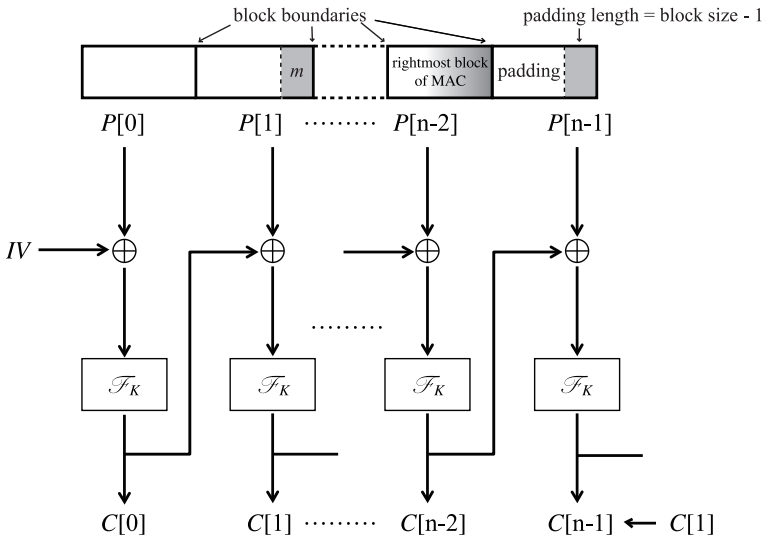


Figure 5: Schematic View of the POODLE Attack

[6]POODLE stands for “Padding Oracle On Downgraded Legacy Encryption”.

An attacker can learn the unknown byte $m$ when a server decrypts the ciphertext without errors (256 times on average). In fact, at the second and the last block, we have

$$\mathscr{F}_K^{-1}(C[1]) = C[0] \oplus P[1] \quad \text{and} \quad \mathscr{F}_K^{-1}(C[n-1]) = C[n-2] \oplus P[n-1],$$

and under the condition that no error is occurred when decrypting, we have

$$P[1] = C[0] \oplus \mathscr{F}_K^{-1}(C[1]) = C[0] \oplus \mathscr{F}_K^{-1}(C[n-1]) = C[0] \oplus C[n-2] \oplus P[n-1].$$

Then

$$\begin{aligned} m &= \text{the last byte of } P[1] = \text{the last byte of } (C[0] \oplus C[n-2] \oplus P[n-1]) \\ &= \text{the last byte of } (C[0] \oplus C[n-2]) \oplus (\text{block size} - 1), \end{aligned}$$

because a server only checks whether or not the last byte of $(\mathscr{F}_K^{-1}(C[1]) \oplus C[n-2])$ equals to the padding length when decrypting.

To avoid the POODLE attack, SSL 3.0 was disabled by default in some web browsers [21, 3, 24] and was then deprecated by IETF RFC 7568 [4].

### 2.3.3 RC4 biases Attacks

RC4 biases attacks are plaintext recovery attacks in the broadcast setting where a same plaintext is encrypted with different keys (See Figure 6). Needless to say, these attacks are not related to the CBC mode. Several biases of the RC4 keystream have already been found (See [18, 1]).

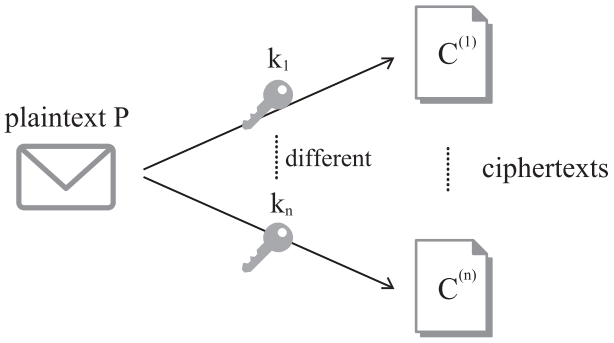


Figure 6: Schematic View of the Broadcast Setting

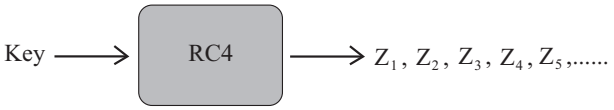


Figure 7: The RC4 Key Stream

For example, in the broadcast setting, second byte of the RC4 keystream (See Figure 7) is biased to zero [23]. In other words, $Z_2 = 0$ occurs with twice the expected probability. So most frequent value of a ciphertext $C_2 (= P_2 \oplus Z_2)$ can be regarded as a plaintext $P_2$ in this case. Given $2^{32}$ ciphertexts, the first 257 bytes of the RC4 key stream $Z_1, Z_2, \ldots, Z_{257}$ can be extracted with the probability more than 0.5 [19].

Note that an attacker does not need multiple receivers and the broadcast setting can be feasible in other attack scenarios.

Because of RC4 biases attacks, IETF RFC 7465 [29] prohibits the use of the RC4 ciphersuites in TLS.

# 3 Preliminaries

## 3.1 Definition

Let $\lambda$ and $\tau$ denote security parameters, where each of them represents the length in byte. The length $\lambda$ is the block cipher's block size, and hence $\lambda$ is a multiple of eight. The negligible function is denoted by $\varepsilon(\lambda)$, or simply by $\varepsilon$.

*Pseudorandom Function and Permutation*: A pseudorandom function (PRF) $\mathscr{P}$ consists of a pair of algorithms $(\mathscr{K},\mathscr{F})$:

- The key generation algorithm $\mathscr{K}$ is a PPT (probabilistic polynomial time) algorithm and generates a key $K$.
- The evaluation algorithm $\mathscr{F}$ is a deterministic polynomial time algorithm. It generates $\mathscr{F}(K,x)$ given the key $K$ and a point $x$.

**Definition 3.1** (Pseudorandom Function, PRF)**.** *We say that $\mathscr{P}=(\mathscr{K},\mathscr{F})$ is PRF if for any* PPT *algorithm A,*

$$\left|\Pr[K \xleftarrow{\$} \mathscr{K} : A^{\mathscr{F}(K,\cdot)}=1]-\Pr[\mathscr{F}' \xleftarrow{\$} \mathscr{R} : A^{\mathscr{F}'(\cdot)}=1]\right| \leq \varepsilon_{\mathrm{PRF}}(\lambda),$$

*where $\mathscr{R}$ is a set of all functions such that both the domain and the range are the same as $\mathscr{F}(K,\cdot)$, respectively.*

If the function $\mathscr{F}_K(\cdot):=\mathscr{F}(K,\cdot)$ is a permutation, then we say that $\mathscr{P}$ is a pseudorandom permutation (PRP). In this case, we denote the negligible function by $\varepsilon_{\mathrm{PRP}}$.

*Symmetric Key Encryption*: The symmetric key encryption (SKE) scheme $\mathscr{SE}$ consists of a triple of algorithms $(\mathscr{K},\mathscr{E},\mathscr{D})$:

- The key generation algorithm $\mathscr{K}$ is a PPT algorithm which generates a key $K$.
- The PPT encryption algorithm $\mathscr{E}$ takes a key $K$ and a plaintext $M$ as input, and outputs a ciphertext $C$. If we consider a *stateful* SKE, then $\mathscr{E}$ has additional input $\mathtt{st}$ as a state, and outputs a new state $\mathtt{st}'$ as well.
- The decryption algorithm $\mathscr{D}$ is a deterministic polynomial time algorithm. This algorithm takes a ciphertext $C$ and a key $K$ as input and outputs a plaintext $M$ or $\perp$ representing an invalid ciphertext. If we consider a stateful SKE then $\mathscr{D}$ is given a state $\mathtt{st}$ and outputs a new state $\mathtt{st}'$ in addition.

The SKE scheme must be "decryptable." That is, for any key $K$ and any plaintext $M$,

$$\mathscr{D}(K,\mathscr{E}(K,M))=M$$

holds.

To define the security, we consider the left-or-right oracle $\mathsf{LR}_{K,b}(M_0,M_1)=\mathscr{E}(K,M_b)$, where $b\in\{0,1\}$.

**Definition 3.2** (IND-CPA). *We say that the SKE $\mathscr{SE} = (\mathscr{K}, \mathscr{E}, \mathscr{D})$ satisfies the $(\varepsilon_{\mathrm{IND}}, q)$ IND-CPA if for any* PPT *algorithm A,*

$$\mathtt{Adv}_{\mathrm{IND}}(\lambda) = \left|\Pr[K \xleftarrow{\$} \mathscr{K}, b \xleftarrow{\$} \{0,1\}, b' \xleftarrow{\$} A^{\mathsf{LR}_{K,b}(\cdot,\cdot)} \mid b = b'] - \frac{1}{2}\right| \leq \varepsilon_{\mathrm{IND}}(\lambda),$$

*where q is the number of queries to* LR *oracle.*

*Message Authentication Code (MAC)*: The message authentication code (MAC) scheme $\mathscr{MA}$ consists of a triple of algorithms $(\mathscr{K}, \mathscr{T}, \mathscr{V})$.

- The key generation algorithm $\mathscr{K}$ is a PPT algorithm and outputs a key $K$.
- The tag generation algorithm $\mathscr{T}$ is a deterministic polynomial-time algorithm. This algorithm takes a key $K$ and a plaintext $M$ as input and outputs a tag $t$ of length $\tau$.
- The verification algorithm $\mathscr{V}$ is a deterministic polynomial-time algorithm. This algorithm takes a key $K$, a message $M$, and a tag $t$ as input, and outputs 0 or 1.

We say that $\mathscr{MA}$ satisfies the completeness if $\mathscr{V}(K,M,t) = 1$ is equivalent to $t = \mathscr{T}(K,M)$. We assume that, for a randomly chosen key $K$, $\mathscr{T}(K,\cdot)$ is a pseudorandom function. The negligible function is denoted as $\varepsilon_{\mathtt{PRF}}$.

### 3.2 The Format in the SSL/TLS

In the CBC mode of the SSL/TLS, before encrypting the plaintext `Content`, some additional information for maintaining the SSL/TLS session is appended. That is, according to the record layer (See Section 2.1),

$$\mathtt{Content}, \mathtt{MAC}, \mathtt{padding}, \mathtt{padding\_length}$$

are encrypted, simultaneously. Here `padding` is a padding, `padding_length` is the length of the padding, and `MAC` is a tag of

$$\mathtt{MAC\_write\_secret}, \mathtt{seq\_num}, \mathtt{type}, \mathtt{version}, \mathtt{length}, \mathtt{fragment}$$

computed by the message authentication code HMAC.

A sequence number `seq_num` is a binary sequence of 64 bit length. This is a counter starting from 0, and it is incremented for each record. This is originally designed for preventing the replay attack, but we show later that this counter makes the "patched" CBC mode in TLS1.0 indistinguishable. There is other information such as `type`, `versions`, etc., but these are not related to our security analysis and we will omit them henceforth.

## 4 The Effect of the Patch to the CBC mode

Let $\lambda$ be a block length of the underlying block cipher (in byte), and let $\|$ be concatenation. Then, for a binary sequence $X$, we define $X[i]$ and $X[i..]$ as

$$X = \overbrace{X[0]}^{\lambda \text{ byte}} \| \overbrace{X[1]}^{\lambda \text{ byte}} \| \cdots \| \overbrace{X[n-1]}^{\leq \lambda \text{ byte}}, \quad X[i..] = \overbrace{X[i]}^{\lambda \text{ byte}} \| \cdots \| \overbrace{X[n-1]}^{\leq \lambda \text{ byte}}.$$

**Algorithm** $\mathscr{K}_{\mathtt{WeakCBC}}$
$K \xleftarrow{\$} \mathscr{K}_{\mathtt{PRP}}$
Output $K$

**Algorithm** $\mathscr{E}_{\mathtt{WeakCBC}}(K, M; \mathtt{st})$
$\mathtt{IV} \leftarrow \mathtt{st}$
$M[0], \ldots, M[n-1] \leftarrow M$
$C[0] \leftarrow \mathscr{F}_{\mathtt{PRP}}(K, M[0] \oplus \mathtt{IV})$
For $i = 1$ to $n-1$
  $C[i] \leftarrow \mathscr{F}_{\mathtt{PRP}}(K, M[i] \oplus C[i-1])$
Output $C = (\mathtt{IV}, C[0], \ldots, C[n-1])$ and $\mathtt{st} = C[n-1]$

Table 1: The original CBC mode in TLS1.0 (WeakCBC mode)

**Algorithm** $\mathscr{K}_{\mathtt{WeakTLS1.0}}$
$K_{\mathtt{WeakCBC}} \xleftarrow{\$} \mathscr{K}_{\mathtt{WeakCBC}}$
$K_{\mathtt{MA}} \xleftarrow{\$} \mathscr{K}_{\mathtt{MA}}$
$K \leftarrow (K_{\mathtt{WeakCBC}}, K_{\mathtt{MA}})$
Output $K$

**Algorithm** $\mathscr{E}_{\mathtt{WeakTLS1.0}}(K, M; \mathtt{st})$
Parse $\mathtt{st}$ as $(\mathtt{st}_{\mathtt{WeakCBC}}, c)$
Parse $K$ as $(K_{\mathtt{WeakCBC}}, K_{\mathtt{MA}})$
$t \leftarrow \mathscr{T}(K_{\mathtt{MA}}, c \| |M| \| M)$
$(C, \mathtt{st}_{\mathtt{WeakCBC}}) \leftarrow \mathscr{E}_{\mathtt{WeakCBC}}(K_{\mathtt{WeakCBC}}, \mathsf{Pad}(M\|t); \mathtt{st})$
Output $(C, (\mathtt{st}_{\mathtt{WeakCBC}}, c + |M|))$

**Algorithm** $\mathscr{D}_{\mathtt{WeakTLS1.0}}(K, C; \mathtt{st})$
Parse $\mathtt{st}$ as $c$
Parse $K$ as $(K_{\mathtt{WeakCBC}}, K_{\mathtt{MA}})$
$M' \leftarrow \mathscr{D}_{\mathtt{WeakCBC}}(K_{\mathtt{WeakCBC}}, C)$
$M'' \leftarrow \mathsf{Pad}^{-1}(M')$
If $M'' \neq \perp$ then parse $M''$ as $M\|t$
else output $\perp$
If $\mathscr{T}(K_{\mathtt{MA}}, c \| |M| \| M) = t$,
  output $(M, c + |M|)$
else output $\perp$

Table 2: Unpatched CBC (WeakTLS1.0)

Hence, except for the last block $X[n-1]$, $X[i]$ is $\lambda$ bytes. Let $X[i]$ be a byte sequence of $\lambda'(\leq \lambda)$ bytes. Then we define $X[i][j]$ and $X[i][j..]$ as

$$X[i] = \overbrace{X[i][0]}^{\text{1 byte}} \| \cdots \| \overbrace{X[i][\lambda'-1]}^{\text{1 byte}}, \quad X[i][j..] = X[i][j] \| \cdots \| X[i][\lambda'-1].$$

## 4.1 Weak CBC Mode in TLS1.0

Let $\mathscr{P} = (\mathscr{K}_{\mathtt{PRP}}, \mathscr{F}_{\mathtt{PRP}})$ be a PRP. The CBC mode in TLS1.0 is implemented as Table 1, where we assume that the length of the message $M$ is a multiple of $\lambda$, and the initial vector $\mathtt{IV}$ is randomly chosen at the beginning. The decryption algorithm $\mathscr{D}_{\mathtt{WeakCBC}}$ is not described since it is trivial.

We call this version of the CBC mode as the WeakCBC mode. Clearly, in the WeakCBC mode, since the adversary knows $\mathtt{IV}(= C[n-1])$ in advance, it does not satisfy the IND-CPA security. This is the reason why the original CBC mode (WeakCBC) is vulnerable to the BEAST attack.

## 4.2 Unpatched CBC

In TLS1.0, the encryption is done by Mac-then-Enc. Hence, the tag is generated before the message is encrypted in the CBC mode. (See Table 2.) In Table 2, $c$ plays the role of the counter which starts from 0. The counter represents the sequence number `seq_num` in Sec.3.2. The other information such as `type` is not related in our security analysis, and hence we will omit it from this algorithm.

| **Algorithm** $\mathscr{K}_{\mathtt{SplTLS1.0}}$ | **Algorithm** $\mathscr{E}_{\mathtt{SplTLS1.0}}(K,M;\mathtt{st})$ |
|---|---|
| $K \xleftarrow{\$} \mathscr{K}_{\mathtt{WeakTLS}}$ | $(C_0,\mathtt{st}) \leftarrow \mathscr{E}_{\mathtt{WeakTLS1.0}}(K,M[0][0];\mathtt{st})$ |
| Output $K$ | If $M$ is one byte then output $(C_0,\mathtt{st})$ |
| | else $(C_1,\mathtt{st}) \leftarrow \mathscr{E}_{\mathtt{WeakTLS1.0}}(K,M[0][1..];\mathtt{st})$ |
| | and output $(C_0,C_1)$ and $\mathtt{st}$ |

Table 3: Patched CBC (SplTLS1.0)

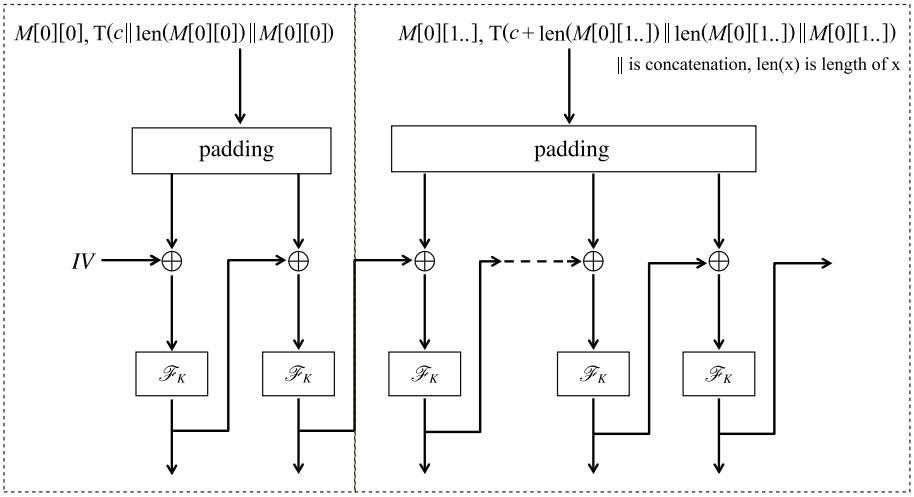


Figure 8: Patched CBC: $1/n-1$ Record Splitting Patch Applied WeakTLS1.0 (SplTLS1.0)

We call the authenticated encryption of Table 2 as WeakTLS1.0. The algorithm $\mathsf{Pad}$ is the padding algorithm which is defined as Eq.(1), and $\mathsf{Pad}^{-1}$ is the algorithm which removes the padding. As defined in Section 3.1, $\mathscr{MA} = (\mathscr{K}_{\mathtt{MA}}, \mathscr{T}, \mathscr{V})$ is the message authentication code.

Since `IV` is predictable, WeakTLS1.0 does not satisfy the IND-CPA property as well.

## 4.3 Patched CBC

To fix security flaws of the CBC mode, some software patches for the WeakTLS1.0 described in Sec.4.2 have been released by browser vendors. For example, a countermeasure to prepend an empty plaintext record before sending the actual plaintext records is no longer used since it is not sufficient for the practical use due to the lack of interconnectivity [25]. At present, the software patch named $1/n-1$ *Record Splitting Patch* [35] is widely used, which is implemented as Table 3, and Figure 8.

We call the authenticated encryption scheme described in Table 3 as SplTLS1.0. For the decryption, the algorithm outputs the plaintexts using $\mathscr{D}_{\mathtt{WeakTLS1.0}}$ multiple times.

In SplTLS1.0, the encryption algorithm for WeakTLS1.0 is invoked two times to encrypt the message $M$. For the first time, the first byte of the message $M[0][0]$ is encrypted, and for the second time the remained message $M[0][1..]$ is encrypted. The security proof of SplTLS1.0 is given as follows:

**Theorem 4.1.** *If $\mathscr{P}$ is PRP, and $\mathscr{MA}$ is (complete) PRF, then SplTLS1.0 satisfies $(\varepsilon_{\mathrm{IND}}, q)$ IND-CPA security, where*

$$\varepsilon_{\mathrm{IND}} = 2\varepsilon_{\mathrm{PRF}} + 2\varepsilon_{\mathrm{PRP}} + \frac{q'(q'-1)}{2^{8\lambda}} + \varepsilon_{\mathrm{G4}} + \frac{q'^2}{2^{8\lambda}}.$$

*and* $8\lambda q'$ *is the bit-length of all the ciphertexts generated by* LR *oracle. For* $\varepsilon_{\mathrm{G4}}$*,*

- *if* $\lambda - 1 \leq \tau$ *then,* $\varepsilon_{\mathrm{G4}} = \frac{q(q-1)}{2^{8\lambda-7}}$
- *else* $\varepsilon_{\mathrm{G4}} = \frac{q(q-1)}{2^{8\tau-1}}$.

The theorem says that the indistinguishability of the patched CBC mode depends on the tag length. For example, if AES and HMAC-SHA1 are used as $\mathscr{P}$ and $\mathscr{MA}$ respectively then $\lambda = 16$, $\tau = 20$ and hence $\varepsilon_{\mathrm{G4}} = q(q-1)/2^{121}$. If the truncated HMAC defined in RFC 6066 [15] is used instead then $\tau = 10$ and hence $\varepsilon_{\mathrm{G4}} = q(q-1)/2^{79}$. $\mathtt{Adv}_{\mathrm{IND}}$ of SplTLS1.0 may increase in this case.

$\mathscr{SE}_{\mathtt{WeakTLS1.0}}$ is not secure against BEAST type of attack, as stated in Section 2.3.1. Then, can we apply BEAST type of attack to the patched CBC mode? We will take a look at a simple example as follows. We will encrypt a plaintext

$$P = r_1||r_2||m \quad \text{where } |r_1| = |m| = 1,\ |r_2| = \lambda - 2$$

by SplTLS1.0 (See Figure 9).

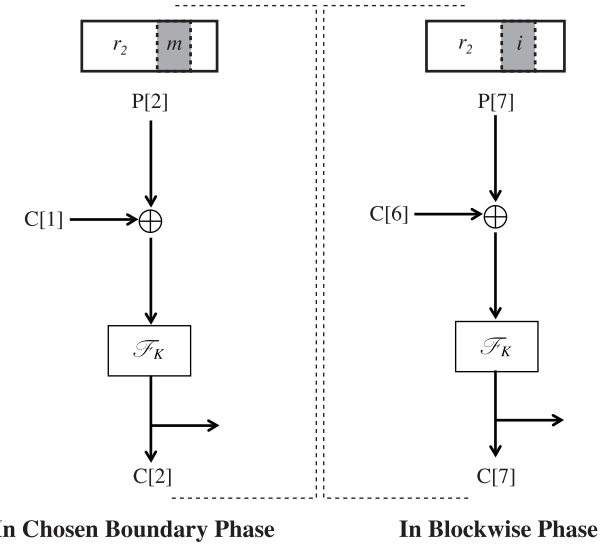


Figure 9: A Simple Example of the Record Splitting

At the first block of the second record in the chosen boundary phase, we have

$$C[1] \oplus \mathscr{F}_K^{-1}(C[2]) = P[2] = r_2||m||(\text{a leftmost byte of a tag in P[2]}).$$

Let us suppose that an attacker makes a target encrypt forged blocks $r_1||r_2||i$ (where $i = 0,\ldots,255$) in the blockwise phase. So, at the first block of the second record in the blockwise phase, we have

$$C[6] \oplus \mathscr{F}_K^{-1}(C[7]) = P[7] = r_2||i||(\text{a leftmost byte of a tag in P[7]}).$$

In this example, we will find the following observations because of the counter $c$ in the tag.

O1. $C[6]$ varies whenever $C[7]$ is computed in the blockwise phase.

O2. Both $P[2]$ and $P[7]$ contain leftmost bytes of different tags respectively.

Because the comparison between $C[2]$ and $C[7]$ is not useful, the patched CBC mode completely defeats all attacker's attempts in the blockwise phase, as was presented in Section 2.3.1. Therefore, they indicate that it is hard to apply BEAST type of attack to the patched CBC mode without enough oracle access for both $\mathscr{P}$ and $\mathscr{MA}$ and then we can not distinguish plaintexts from these ciphertexts.

# 5 Security Proof of Theorem 4.1

We define a sequence of games and prove its IND-CPA security. In Game $i$, the probability of the adversary $D$ outputting 1 is described by

$$\Pr[D = 1 \mid \text{Game } i].$$

Game 0: In this game, we set $b = 0$ in the definition of IND-CPA. Therefore,

$$\Pr[D = 1 \mid \text{Game } 0].$$

Game 1: This is the same as Game 0 except for the following. The modification is to replace the PRP $\mathscr{F}$ with the random permutation. By the definition of PRP,

$$|\Pr[D = 1 \mid \text{Game } 0] - \Pr[D = 1 \mid \text{Game } 1]| \leq \varepsilon_{\mathrm{PRP}}.$$

Game 2: This game is the same as Game 1 except for the following. We replace the random permutation $\mathscr{P}$ with the random function. By the switching lemma of [7],

$$|\Pr[D = 1 \mid \text{Game } 1] - \Pr[D = 1 \mid \text{Game } 2]| \leq \frac{q'(q'-1)}{2^{8\lambda+1}},$$

where $q'$ is the number of queries to the random permutation. Therefore, this is a total block length of the ciphertexts.

Game 3: This is the same as Game 2 except for the following. We replace $\mathscr{MA}$ modeled as the PRF with the random function. Since the difference is bounded by the definition of the PRF,

$$|\Pr[D = 1 \mid \text{Game } 2] - \Pr[D = 1 \mid \text{Game } 3]| \leq \varepsilon_{\mathrm{PRF}}.$$

Game 4: This game is the same as Game 3 except for the following. Let $M_i$ be the $i$-th message to be encrypted in $\mathsf{LR}$ oracle, and let $c_i$ be its counter. Also we define

$$I_i = \mathsf{Pad}(M_i[0][0] \| \mathscr{T}(c_i \| |M_i[0][0]| \| |M_i[0][0])).$$

In this game, if there exists a pair $(i, j)$ $(i \neq j)$ such that $I_i[0] = I_j[0]$ then $\mathsf{LR}$ oracle stops. Let $\mathtt{Coll}_{i,j}$ be the event that there exists a pair $(i, j)$ $(i \neq j)$ such that $I_i[0] = I_j[0]$. Then, if for every $i, j$ $(i \neq j)$, $\mathtt{Coll}_{i,j}$ does not occur then the probability that $D$ outputs 1 in Game 3 and in Game 4 are the same.

Let us estimate the amount of $\Pr[\mathtt{Coll}_{i,j}]$. Depending on the length of the tag $\tau$, we consider two cases $\lambda - 1 \leq \tau$ and $\lambda - 1 > \tau$.

Case $\lambda - 1 \leq \tau$ in Game 4: Since $M[0][0]$ is 1 byte which can be controlled by the adversary, and $\lambda - 1 \leq \tau$, the input to $\mathscr{F}_K$ is

$$A = \mathtt{IV} \oplus M[0][0] \| t[0][0] \| \cdots \| t[0][\lambda - 2],$$

where $t = \mathscr{T}(c_i \| |M_i[0][0]| \| M_i[0][0])$.

Further, since $c_i$ is a counter, the input $c_i \| |M_i[0][0]| \| M_i[0][0]$ to $\mathscr{T}$ is not duplicate. Hence, $A[0][1..]$ is random since $\mathscr{T}$ is a random function. Therefore, for every $i, j$ $(i \neq j)$, $\Pr[\mathtt{Coll}_{i,j}] \leq 1/2^{8\lambda-8}$. Taking the union bound, we have

$$|\Pr[D = 1 \mid \text{Game 3}] - \Pr[D = 1 \mid \text{Game 4}]| \leq \varepsilon_{\text{G4}}$$

where $\varepsilon_{\text{G4}} = \frac{q(q-1)}{2^{8\lambda-7}}$, and $q$ is the number of queries to LR oracle.

Case $\lambda - 1 > \tau$ in Game 4: By the similar discussion as above, we can estimate the difference as

$$|\Pr[D = 1 \mid \text{Game 3}] - \Pr[D = 1 \mid \text{Game 4}]| \leq \varepsilon_{\text{G4}},$$

where $\varepsilon_{\text{G4}} = \frac{q(q-1)}{2^{8\tau-1}}$.

Game 5: This game is the same as Game 4 except for $b = 1$. We prove that the difference of probability $D$ outputting 1 in Game 5 and in Game 4 is

$$|\Pr[D = 1 \mid \text{Game 4}] - \Pr[D = 1 \mid \text{Game 5}]| \leq \frac{q'^2}{2^{8\lambda}}. \tag{2}$$

If the input to the random function $\mathscr{F}_K$ is not duplicate, then a bit $b$ is information theoretically hidden. Therefore, we estimate the probability that the input to $\mathscr{F}_K$ duplicates. Let Bad be the event that input to the random function duplicates. Then, the left-hand side of inequality (2) is bounded by $\Pr[\mathsf{Bad}]$.

The oracle LR encrypts $M_0$ or $M_1$. From the previous game we know that the first query $I_i[0]$ to $\mathscr{F}_K$ is not duplicated. Hence, we can estimate the probability of Bad happens as

$$\Pr[\mathsf{Bad}] \leq \frac{q+1}{2^{8\lambda}} + \frac{q+2}{2^{8\lambda}} + \cdots + \frac{q'}{2^{8\lambda}} \leq \frac{q'^2}{2^{8\lambda}}$$

Game 6: This game is the same as Game 5 except for the followings. Firstly we replace the random function $\mathscr{T}$ in $\mathscr{MA}$ with the PRF, and then replace the random function $\mathscr{F}_K$ with the PRP. Since this modification implies going the reverse direction in the sequence of games, we have

$$\begin{aligned} &|\Pr[D = 1 \mid \text{Game 5}] - \Pr[D = 1 \mid \text{Game 6}]| \\ &\leq \varepsilon_{\text{PRF}} + \varepsilon_{\text{PRP}} + \frac{q'(q'-1)}{2^{8\lambda+1}}. \end{aligned}$$

Since Game 0 is $b = 0$ in the game IND-CPA and Game 6 is $b = 1$ in the game IND-CPA, we have

$$\begin{aligned} |\Pr[K \xleftarrow{\$} \mathscr{K}_{\mathtt{SplTLS1.0}}, b \xleftarrow{\$} \{0,1\}, b' \xleftarrow{\$} A^{\mathsf{LR}_{K,b}(\cdot,\cdot)} \mid b = b'] - \frac{1}{2}| \\ \leq 2\varepsilon_{\text{PRF}} + 2\varepsilon_{\text{PRP}} + \frac{q'(q'-1)}{2^{8\lambda}} + \varepsilon_{\text{G4}} + \frac{q'^2}{2^{8\lambda}}, \end{aligned}$$

where if $\lambda - 1 \leq \tau$ then, $\varepsilon_{\text{G4}} = \frac{q(q-1)}{2^{8\lambda-7}}$, and else $\varepsilon_{\text{G4}} = \frac{q(q-1)}{2^{8\tau-1}}$. This concludes the proof.

# 6 Conclusion

We have given proofs that the patched CBC mode which is currently recommended by major browser vendors satisfies indistinguishability. As presented by Möller, security flaws of the CBC mode were previously found by several experts. The patched CBC mode has arisen because the countermeasure to prepend an empty plaintext record fragment that is compliant with the TLS1.0 specification did not satisfy interoperability between some devices. The security is guaranteed if the length of the tag is longer than the block length of the underlying block cipher. We have also shown that if the tag length is shorter than the block length then the security bound may not be tight enough for the real use. In spite of our securiy proof, we generally recommend using the latest version of TLS.

# References


[1] N. J. AlFardan, D. J. Bernstein, K. G. Paterson, B. Poettering, and J. C. N. Schuldt. On the Security of RC4 in TLS. In *Proc. of the 22th USENIX Security Symposium, Washington, DC, USA*, pages 305–320, August 2013.

[2] N. J. AlFardan and K. G. Paterson. Lucky Thirteen: Breaking the TLS and DTLS Record Protocols. In *Proc. of the 2013 IEEE Symposium on Security and Privacy (SP'13), Berkeley, California, USA*, pages 526–540. IEEE, May 2013.

[3] R. Barnes. The POODLE Attack and the End of SSL 3.0, October 2014. [Online; Accessed on February 20, 2016] `https://blog.mozilla.org/security/2014/10/14/the-poodle-attack-and-the-end-of-ssl-3-0/`.

[4] R. Barnes, M. Thomson, , A. Pironti, and A. Langley. Deprecating Secure Sockets Layer Version 3.0. IETF RFC 7568, June 2015. `http://www.ietf.org/rfc/rfc7568.txt`.

[5] A. Barth. HTTP State Management Mechanism. IETF RFC 6265, April 2011. `http://www.ietf.org/rfc/rfc6265.txt`.

[6] A. Barth. The Web Origin Concept. IETF RFC 6454, December 2011. `http://www.ietf.org/rfc/rfc6454.txt`.

[7] M. Bellare and P. Rogaway. The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In *Advances in Cryptology - EUROCRYPT 2006, Proc. of the 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'06), St. Petersburg, Russia*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer Berlin Heidelberg, May-June 2006.

[8] B. Canvel, A. P. Hiltgen, S. Vaudenay, and M. Vuagnoux. Password Interception in a SSL/TLS Channel. In *Advances in Cryptology - CRYPTO 2003, Proc. of the 23rd Annual International Cryptology Conference (CRYPTO'03), Santa Barbara, California, USA*, volume 2729 of *Lecture Notes in Computer Science*, pages 583–599. Springer Berlin Heidelberg, August 2003.

[9] T. Dierks and C. Allen. The TLS Protocol Version 1.0. IETF RFC 2246, January 1999. `http://www.ietf.org/rfc/rfc2246.txt`.

[10] T. Dierks and E. Rescorla. The TLS Protocol Version 1.1. IETF RFC 4346, April 2006. `http://www.ietf.org/rfc/rfc4346.txt`.

[11] T. Dierks and E. Rescorla. The TLS Protocol Version 1.2. IETF RFC 5246, August 2008. `http://www.ietf.org/rfc/rfc5246.txt`.

[12] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[13] T. Duong and J. Rizzo. Here Come The $\oplus$ Ninjas, May 2011. [Online; Accessed on February 20, 2016] `http://netifera.com/research/beast/beast_DRAFT_0621.pdf`.

[14] M. Dworkin. Recommendation for Block Cipher Modes of Operation, Methods and Techniques. NIST Special Publication 800-38A, December 2001. [Online; Accessed on February 20, 2016] `http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf`.

[15] D. Eastlake. Transport Layer Security (TLS) Extensions: Extension Definitions. IETF RFC 6066, January 2011. http://www.ietf.org/rfc/rfc6066.txt.

[16] A. O. Freier, P. Karlton, and P. C. Kocher. The Secure Sockets Layer (SSL) Protocol Version 3.0. IETF RFC 6101, August 2011. http://www.ietf.org/rfc/rfc6101.txt.

[17] P. Gutmann. Encrypt-then-MAC for Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). IETF RFC 7366, September 2014. http://www.ietf.org/rfc/rfc7366.txt.

[18] T. Isobe, T. Ohigashi, Y. Watanabe, and M. Morii. Full Plaintext Recovery Attack on Broadcast RC4. In *Fast Software Encryption - Revised Selected Papers of the 20th International Workshop (FSE'13), Singapore*, volume Lecture Notes in Computer Science of *8424*, pages 179–202. Springer Berlin Heidelberg, March 2013.

[19] T. Isobe, T. Ohigashi, Y. Watanabe, and M. Morii. Comprehensive Analysis of Initial Keystream Biases of RC4. *IEICE Transactions*, 97-A(1):139–151, 2014.

[20] T. Kurokawa, R. Nojima, and S. Moriai. Can We Securely Use CBC Mode in TLS1.0? In *Information and Communication Technology, Proc. of the 3rdThird IFIP TC 5/8 International Conference, ICT-EurAsia 2015, and 9th IFIP WG 8.9 Working Conference, CONFENIS 2015, Held as Part of WCC 2015, Daejeon, Korea*, volume 9357 of *Lecture Notes in Computer Science*, pages 151–160, October 2015.

[21] A. Langley. An update on SSLv3 in Chrome, October 2014. [Online; Accessed on February 20, 2016] https://groups.google.com/a/chromium.org/forum/#!topic/security-dev/Vnhy9aKM_l4.

[22] A. Langley. The POODLE bites again (08 Dec 2014), December 2014. [Online; Accessed on February 20, 2016] https://www.imperialviolet.org/2014/12/08/poodleagain.html.

[23] I. Mantin and A. Shamir. A Practical Attack on Broadcast RC4. In *Fast Software Encryption, Revised Papers of the 8th Fast Software Encryption (FSE'01) Yokohama, Japan*, volume 2355 of *Lecture Notes in Computer Science*, pages 152–164. Springer Berlin Heidelberg, April 2001.

[24] Microsoft Security Response Center (MSRC). Security Advisory 3009008 updated, October 2014. [Online; Accessed on February 20, 2016] http://blogs.technet.com/b/msrc/archive/2014/10/29/security-advisory-3009008-released.aspx.

[25] B. Möller. Security of CBC Ciphersuites in SSL/TLS: Problems and Countermeasures, May 2004. [Online; Accessed on February 20, 2016] http://www.openssl.org/~bodo/tls-cbc.txt.

[26] B. Möller, T. Duong, and K. Kotowicz. This POODLE Bites: Exploiting The SSL 3.0 Fallback, September 2014. [Online; Accessed on February 20, 2016] https://www.openssl.org/~bodo/ssl-poodle.pdf.

[27] National Institute of Standards and Technology. Specification for the ADVANCED ENCRYPTION STANDARD (AES). Federal Information Processing Standards Publication (FIPS) 197, November 2001. [Online; Accessed on February 20, 2016] http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[28] National Institute of Standards and Technology. The Keyed-Hash Message Authentication Code (HMAC). Federal Information Processing Standards Publication (FIPS) 198-1, July 2008. [Online; Accessed on February 20, 2016] http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf.

[29] A. Popov. Prohibiting RC4 Cipher Suites. IETF RFC 7465, February 2015. http://www.ietf.org/rfc/rfc7465.txt.

[30] J. Rizzo and T. Duong. The CRIME attack, September 2012. [Online; Accessed on February 20, 2016] Ekoparty 2012, https://docs.google.com/presentation/d/11eBmGiHbYcHR9gL5nDyZChu_-lCa2GizeuOfaLU2HOU/.

[31] P. Rogaway. Problems with Proposed IP Cryptography, April 1995. [Online; Accessed on February 20, 2016] http://www.cs.ucdavis.edu/~rogaway/papers/draft-rogaway-ipsec-comments-00.txt.

[32] RSA Laboratory. PKCS #1 v2.2: RSA Cryptography Standard. RSA Laboratories' Public-Key Cryptography Standards (PKCS), October 2012. [Online; Accessed on February 20, 2016] https://www.emc.com/collateral/white-papers/h11300-pkcs-1v2-2-rsa-cryptography-standard-wp.pdf.

[33] P. G. Sarkar and S. Fitzgerald. ATTACKS ON SSL - A Comprehensive Study of BEAST, CRIME, TIME, BREACH, LUCKY 13 & RC4 BIASES, August 2013. [Online; Accessed on February 20, 2016] https://www.nccgroup.trust/globalassets/our-research/us/whitepapers/ssl_attacks_survey.pdf.

[34] B. Schneier. *Applied Cryptography - Protocols, Algorithms, and Source Code in C (2nd. edition)*. Wiley, 1996.

[35] X. Su. Bugzilla Bug 665814 Comment 59, July 2011. [Online; Accessed on February 20, 2016] `https://bugzilla.mozilla.org/show_bug.cgi?id=665814#c59`.

[36] Trustworthy Internet Movement. SSL Pulse - Survey of the SSL Implementation of the Most Popular Web Sites. [Online; Accessed on February 20, 2016] `https://www.trustworthyinternet.org/ssl-pulse/`.

[37] S. Vaudenay. Security Flaws Induced by CBC Padding - Applications to SSL, IPSEC, WTLS ... In *Advances in Cryptology - EUROCRYPT 2002, Proc. of the 2002 International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'02), Amsterdam, The Netherlands*, volume 2332 of *Lecture Notes in Computer Science*, pages 534–546. Springer Berlin Heidelberg, April-May 2002.

---

## Author Biography

**Takashi Kurokawa** received the B.S. degree in mathematics from Waseda University, Tokyo, Japan, in 1988 and the M.S. in mathematics from the University of Tokyo in 1991. He works for NICT as a guest researcher and a technical expert since 2004. He is taking a doctor's course at Graduate School of Engineering and Resource Science, Akita University. His research interests include information security, especially security evaluations of cryptographic technologies.

**Ryo Nojima** is a senior researcher at the National Institute of Information and Communication Technology. His main research interests include cryptography and information security, especially McEliece cryptosystems and SSL/TLS.

**Shiho Moriai** received the B.E. degree from Kyoto University in 1993, and Ph.D. from the University of Tokyo in 2003. She is Director of Security Fundamentals Laboratory, Network Security Research Institute, NICT. She serves on several committees for information security of Japan e-government systems and critical infrastructure systems.