Can We Securely Use CBC Mode in TLS1.0?

Takashi Kurokawa, Ryo Nojima, Shiho Moriai

Can We Securely Use CBC Mode in TLS1.0?

Takashi Kurokawa, Ryo Nojima and Shiho Moriai

National Institute of Information and Communications Technology (NICT),
4-2-1, Nukui-Kitamachi, Koganei, Tokyo 184-8795, Japan
{blackriver,ryo-no,shiho.moriai}@nict.go.jp

Abstract. Currently, TLS1.0 is one of the most widely deployed protocol versions for SSL/TLS. In TLS1.0, there are only two choices for the bulk encryption, i.e., RC4 or block ciphers in the CBC mode, which have been criticized to be insecure.
In this paper, we explore the current status of the CBC mode in TLS1.0 and prove theoretically that the current version of the (patched) CBC mode in TLS1.0 satisfies *indistinguishability*, which implies that it is secure against BEAST type of attacks.

Keywords: TLS1.0, The BEAST attack, Security.

1 Introduction

1.1 CBC Mode in TLS1.0

The SSL/TLS is one of the most widely deployed cryptographic protocols used in the network. In fact, SSL/TLS is employed in almost all the popular services for online shopping and online banking. At the same time, many cryptographic attacks against SSL/TLS have been found, e.g., CRIME, Lucky Thirteen [2], BEAST [5], POODLE [7] and RC4 bias attacks [1, 6].[1]

In SSL/TLS, many cryptographic primitives have been employed, e.g., RSA, DH(E), AES, RC4, CBC mode, and HMAC. Among them, we are going to focus on the *CBC mode* in *TLS 1.0*, which is one of the most problematic cryptographic primitives in SSL/TLS. To see this, let us introduce how the CBC mode is used in SSL/TLS. In SSL/TLS, a plaintext is "tagged" before the encryption. That is, to encrypt a plaintext M, the tag t is firstly generated and then the message

$$M' = M \| t$$

is encrypted by the CBC mode. Then, the ciphertext of the (tagged) message $M' = (M'[0], M'[1], \ldots, M'[m-1])$ is encrypted as

$$\mathtt{IV}, \mathcal{F}_K(\mathtt{IV} \oplus M'[0]), \ldots, \mathcal{F}_K(C[m-2] \oplus M'[m-1] \| \mathtt{PAD} \| \mathtt{PAD\_LEN}), \qquad (1)$$

where $\mathcal{F}_K : \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a block cipher modeled as the pseudorandom permutation, λ is the block length, $\mathtt{IV}$ is an initial vector, $\mathtt{PAD}$ is a padding,

[1] For the overview of the recent attacks, see [9].

$\mathtt{PAD\_LEN}$ is the length of $\mathtt{PAD}$, $C[0] = \mathcal{F}(\mathtt{IV} \oplus M[0])$ and $C[i] = \mathcal{F}_K(C[i-1] \oplus M'[i])$ for $1 \leq i \leq m-1$.

The CBC mode in TLS1.0 has two potential weaknesses: one is in the padding and the other is in the choice of the initial vector `IV` [13].

Padding: In the encryption of the form Eq.(1), which is known as Mac-then-Enc, the message authentication code is not applied to the padding. That is, the padding is appended after the generation of the tag. Accordingly, we can consider two errors: the error of the padding and that of the message authentication code. If the adversary can distinguish these two errors, an attack known as the *padding oracle attack* [10] works. For a concrete example, there exists a timing analysis [4] which enables the adversary to distinguish these two errors. However, this problem has been repaired in some implementations of SSL/TLS, e.g., OpenSSL `0.9.6c`, `0.9.6i`, and `0.9.7a`. There is a possibility that other side channel information can be used to attack the CBC mode. In fact, for SSL3.0, the Möller et al. [7] showed a practical attack against the CBC mode in SSL3.0, named the POODLE attack. However, this attack cannot be applied directly to the CBC mode in TLS1.0 since a different padding scheme is employed.

Choice of `IV`: In TLS1.0, the initial vector `IV` 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 [8]. Since this means that `IV` is predictable from the adversary's viewpoint, the CBC mode in TLS1.0 does not satisfy indistinguishability.

However, this does not immediately imply that the adversary can recover the whole plaintext and moreover it was expected that the time complexity of the recovering the plaintext would be $O(2^{\lambda})$ for one block of ciphertexts. Unfortunately, such an idea was not true. Duong and Rizzo demonstrated the BEAST attack [5] whose time complexity is $O(\lambda)$.

1.2 On BEAST Attack

To launch the BEAST attack, two underlying conditions must be satisfied. One is that there exists a software bug on Same Origin Policy (SOP) in the browser and the other is the predictability of IV, which is the case of the CBC mode in TLS1.0. The attack has huge impact since Duang and Rizzo found the software bug on SOP in Java. At present, a software patch for Java is released but there is a possibility that there are many software bugs. Hence, browser vendors such as Microsoft, and Mozilla released a software patch for the CBC mode in addition to the patch for Java [9].

1.3 Contributions

According to [14], currently, TLS1.0 is the most widely deployed protocol version in SSL/TLS, and the CBC mode is used in many ciphersuites. Although the software patch is released for the CBC mode, there has been a problem remained. That is, it is not clarified whether or not the patched CBC mode is secure against BEAST type of attacks. In this paper, we show that the patched CBC mode satisfies the indistinguishability, which implies that the CBC mode is secure against BEAST type of attacks. As far as we know, this is the first time to show that the current version of the CBC mode in the TLS1.0 satisfies the indistinguishability despite the fact that TLS1.0 is widely used in practice.

2 Preliminaries

2.1 Definition

Let λ, τ denote security parameters, where each of them represents the length in byte. The length is often considered in byte, and hence λ, τ are multiple of eight. The negligible function is denoted by $\epsilon(\lambda)$, or simply by ϵ.

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

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

Definition 1 (Pseudorandom Function, PRF). *We say that $\mathcal{P} = (\mathcal{K}, \mathcal{F})$ is PRF if for any* PPT *algorithm A,*

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

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

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

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

- The key generation algorithm $\mathcal{K}$ is a PPT algorithm which generates a key K.
- The PPT encryption algorithm $\mathcal{E}$ takes a key K and a plaintext M as input, and outputs a ciphertext C. If we consider a *stateful* SKE, then $\mathcal{E}$ has additional input $\mathtt{st}$ as a state, and outputs a new state $\mathtt{st}'$ as well.
- The decryption algorithm $\mathcal{D}$ 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 $\mathcal{D}$ 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,

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

must be satisfied.

To define the security, we consider the function $\mathsf{LR}_{K,b}(M_0, M_1) = \mathcal{E}(K, M_b)$, where $b \in \{0, 1\}$.

Definition 2 (IND-CPA). *We say that the SKE $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ satisfies the $(\epsilon_{\mathrm{IND}}, q)$ IND-CPA if for any* PPT *algorithm A,*

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

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

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

- The key generation algorithm $\mathcal{K}$ is a PPT algorithm and outputs a key K.
- The tag generation algorithm $\mathcal{T}$ 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 τ.
- The verification algorithm $\mathcal{V}$ 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 $\mathcal{MA}$ satisfies the completeness if $\mathcal{V}(K, M, t) = 1$ is equivalent to $t = \mathcal{T}(K, M)$. We assume that, for a randomly chosen key K, $\mathcal{T}(K, \cdot)$ is a pseudorandom function. The negligible function will be denoted as $\epsilon_{\mathtt{PRF}}$.

2.2 The Format in SSL/TLS

In the CBC mode of SSL/TLS, to encrypt the plaintext `CONTENT`, some additional information for maintaining the SSL/TLS session is appended. That is,

$$\mathtt{CONTENT}, \mathtt{MAC}, \mathtt{PAD}, \mathtt{PAD\_LEN}$$

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

$$\mathtt{SEQ\_NUM}, \mathtt{CONTENT\_TYPE}, \mathtt{LEN}, \mathtt{CONTENT}$$

generated by the message authentication code HMAC.

A sequence number `SEQ_NUM` is a binary sequence of length 64 in bit. This is a counter starting from 0, and the length of the message `CONTENT` is incremented for every encryption. This is originally 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 `CONTENT_TYPE`, but these are not related to our security analysis.

```
Algorithm $\mathcal{K}_{\texttt{WeakCBC}}$
    $K \xleftarrow{\$} \mathcal{K}_{\texttt{PRP}}$
    Output $K$
```

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

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

3 The Effect of the Patch

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

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

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

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

3.1 Weak CBC Mode in TLS1.0

Let $\mathcal{P} = (\mathcal{K}_{\texttt{PRP}}, \mathcal{F}_{\texttt{PRP}})$ 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 multiple of λ, and the initial vector $\texttt{IV}$ is chosen random at the beginning. The decryption algorithm $\mathcal{D}_{\texttt{WeakCBC}}$ 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 $\texttt{IV}(= C[n-1])$ 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.

3.2 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 $\texttt{SEQ\_NUM}$ in Sec.2.2. Other information such as $\texttt{TYPE}$ is not related in our security analysis, and hence we remove from this algorithm.

The algorithm Pad is the padding algorithm which is defined as Eq.(1), and Pad^{-1} is the algorithm which removes the padding.

```
Algorithm $\mathcal{K}_{\texttt{WeakTLS1.0}}$
    $K_{\texttt{WeakCBC}} \xleftarrow{\$} \mathcal{K}_{\texttt{WeakCBC}}$
    $K_{\texttt{MA}} \xleftarrow{\$} \mathcal{K}_{\texttt{MA}}$
    $K \leftarrow (K_{\texttt{WeakCBC}}, K_{\texttt{MA}})$
    Output $K$

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

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

Table 2. Unpatched CBC (WeakTLS1.0)

```
Algorithm $\mathcal{K}_{\texttt{SplTLS1.0}}$
    $K \xleftarrow{\$} \mathcal{K}_{\texttt{WeakTLS}}$
    Output $K$

Algorithm $\mathcal{E}_{\texttt{SplTLS1.0}}(K, M; \texttt{st})$
    $(C_0, \mathsf{st}) \leftarrow \mathcal{E}_{\texttt{WeakTLS1.0}}(K, M[0][0]; \texttt{st})$
    If $M$ is one byte then output $(C_0, \mathsf{st})$
    else $(C_1, \mathsf{st}) \leftarrow \mathcal{E}_{\texttt{WeakTLS1.0}}(K, M[0][1..]; \mathsf{st})$
    and output $(C_0, C_1)$ and $\texttt{st}$
```

Table 3. Patched CBC (SplTLS1.0)

Note that $\mathcal{MA} = (\mathcal{K}_{\texttt{MA}}, \mathcal{T}, \mathcal{V})$ is the message authentication code. We say that the authenticated encryption of Table 2 as WeakTLS1.0.

Since $\texttt{IV}$ is predictable, WeakTLS1.0 does not satisfy the IND-CPA property as well.

3.3 Patched CBC

By the BEAST attack, some software patches for the WeakTLS1.0 described in Sec.3.2 are released by browser vendors. Since some patches are not sufficient for the practical use due to the lack of the interconnectivity, they are no longer used. At present, the software patch named $1/n-1$ *Record Splitting Patch* [11] is widely used, which is implemented as Table 3, and Figure 1. We call the authenticated encryption scheme described in Table 3 as SplTLS1.0. For the decryption, the algorithm outputs the plaintexts using $\mathcal{D}_{\texttt{WeakTLS1.0}}$ 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:

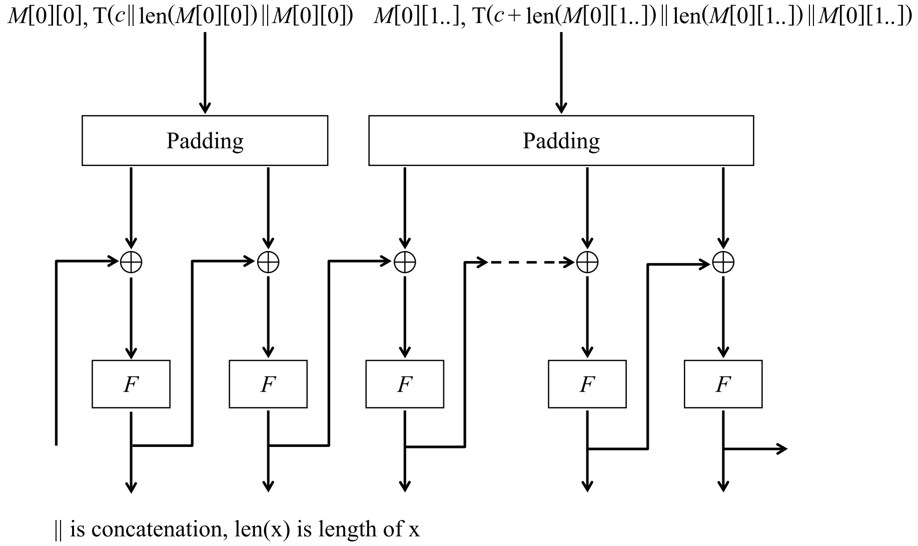
M[0][0], T(c|| len(M[0][0]) ||M[0][0])
M[0][1..], T(c+ len(M[0][1..]) || len(M[0][1..]) ||M[0][1..])
Padding
Padding
F
F
F
F
F
|| is concatenation, len(x) is length of x

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

Theorem 1. *If $\mathcal{P}$ is PRP, and $\mathcal{MA}$ is (complete) PRF, then SplTLS1.0 satisfies $(\epsilon_{\mathrm{IND}}, q)$ IND-CPA security, where*

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

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

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

Therefore, the indistinguishability of the patched CBC mode depends on the tag length. For example, if AES and HMAC-SHA1 are employed then $\lambda = 16$, $\tau = 20$ and hence $\epsilon_{\mathrm{G4}} = q(q-1)/2^{121}$. However, if the truncated message authenticated code is used instead (as RFC 6066 [15]), then $\tau = 10$ and hence $\epsilon_{\mathrm{G4}} = q(q-1)/2^{79}$.

4 Security Proof of Theorem 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 $\mathcal{F}$ with the random permutation. By the definition of PRP,

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

Game 2: This game is the same as Game 1 except for the following. We replace the random permutation $\mathcal{P}$ with the random function. By the switching lemma of [3],

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

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 $\mathcal{MA}$ modeled as the PRF with the random function. Since the difference is bounded by the definition of the PRF,

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

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 LR oracle, and let c_i be its counter. Also we define

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

In this game, if there exists a pair (i, j) $(i \neq j)$ such that $I_i[0] = I_j[0]$ then 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 τ, 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 $\mathcal{F}_K$ is

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

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

Further, since c_i is a counter, the input $c_i \| |M_i[0][0]| \| M_i[0][0]$ to $\mathcal{T}$ is not duplicate. Hence, $A[0][1..]$ is random since $\mathcal{T}$ 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

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

where $\epsilon_{\text{G4}} = \frac{q(q-1)}{2^{8\lambda-7}}$, 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

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

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

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 $\mathcal{F}_K$ is not duplicate, then a bit b is information theoretically hidden. Therefore, we estimate the probability that the input to $\mathcal{F}_K$ 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 $\mathcal{F}_K$ 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 $\mathcal{T}$ in $\mathcal{MA}$ with the PRF, and then replace the random function $\mathcal{F}_K$ with the PRP. Since this modification implies going the reverse direction in the sequence of games, we have

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

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

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

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

5 Conclusion

We have proved that the patched CBC mode which is currently recommended by major browser vendors satisfies indistinguishability. The security is guaranteed if the length of the tag is longer than the block length of the underlying block cipher. However, there are some situations that the tag length τ is shorter than the block length λ. For example, the truncated HMAC defined in RFC 6066 [15] uses the short tag. In this special case, the security bound is not tight enough for the real use.

References

1. N.J. AlFardan, D.J. Bernstein, K.G. Paterson, B. Poettering, J.C.N. Schuldt, "On the Security of RC4 in TLS and WPA," USENIX Security Symposium 2013.
2. N.J. AlFardan, K.G. Paterson, "Lucky Thirteen: Breaking the TLS and DTLS Record Protocols," IEEE Symposium on Security and Privacy 2013, pp.526–540.
3. M. Bellare, P. Rogaway, "The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs," EUROCRYPT 2006, pp.409–426.
4. B. Canvel, A.P. Hiltgen, S. Vaudenay, M. Vuagnoux, "Password Interception in a SSL/TLS Channel," CRYPTO 2003, pp.583–599.
5. T. Duong, J. Rizzo, "Here Come The ⊕ Ninjas," 2011. http://nerdoholic.org/uploads/dergln/beast_part2/ssl_jun21.pdf
6. T. Isobe, T. Ohigashi, Y. Watanabe, M. Morii "Full Plaintext Recovery Attack on Broadcast RC4," FSE2013.
7. B. Möller, T. Duong, K. Kotowicz. "This POODLE Bites: Exploiting The SSL 3.0 Fallback," https://www.openssl.org/~bodo/ssl-poodle.pdf
8. P. Rogaway, "Problems with Proposed IP Cryptography," 1995. http://www.cs.ucdavis.edu/~rogaway/papers/draft-rogaway-ipsec-comments-00.txt
9. P.G. Sarkar, S. Fitzgerald, "Attacks on SSL a Comprehensive Study of BEAST, CRIME, TIME, BREACH, LUCKY 13 & RC4 BIASES," 2013. https://www.isecpartners.com/media/106031/ssl_attacks_survey.pdf
10. S. Vaudenay, "Security Flaws Induced by CBC Padding - Applications to SSL, IPSEC, WTLS ...," EUROCRYPT 2002, pp.534–546.
11. "Bug 665814," https://bugzilla.mozilla.org/show_bug.cgi?id=665814#c59
12. "Information Security, Is BEAST really fixed in all modern browsers?," http://security.stackexchange.com/questions/18505/is-beast-really-fixed-in-all-modern-browsers
13. "Security of CBC Ciphersuites in SSL/TLS: Problems and Countermeasures," http://www.openssl.org/~bodo/tls-cbc.txt
14. "SSL Pulse, Survey of the SSL Implementation of the Most Popular Web Sites," https://www.trustworthyinternet.org/ssl-pulse/
15. Transport Layer Security (TLS) Extensions: Extension Definitions, http://tools.ietf.org/html/rfc6066