Exploiting TLS to disrupt privacy of traffic in web-application

Student Name: Sandipan Biswas

IIIT-D-MTech-CS-IS-12-018
May 2^{nd}, 2014

Indraprastha Institute of Information Technology
New Delhi

Thesis Committee
Dr. Somitra Sanadhya (Chair)
Dr. Shweta Agrawal (IIT Delhi)
Dr. Debajyoti Bera (IIIT Delhi)

Submitted in partial fulfillment of the requirements
for the Degree of M.Tech. in Computer Science,
with specialization in Information Security

Keywords: Side Channel Attack, TLS , Padding, privacy , WPA2 , k-indistinguishability

Certificate

This is to certify that the thesis titled **"Exploting TLS to disrupt privacy of web-application's traffic "** submitted by **Sandipan Biswas** for the partial fulfilment of the requirements for the degree of *Master of Technology* in *Computer Science & Engineering* is a record of the bonafide work carried out by him under our guidance and supervision in the Security and Privacy group at Indraprastha Institute of Information Technology, Delhi. This work has not been submitted anywhere else for the reward of any other degree.

Dr. Donghoon Chang
IIIT Delhi

Dr. Somitra Sanadhya
IIIT Delhi

Abstract

The Transport Layer Protocol (TLS) ensures confidentiality and integrity of traffic between communicating parties over internet. Almost all web applications commonly use TLS. A block cipher (such as AES, Camellia etc.) is used in a mode of operation (such as CBC, GCM etc.) to achieve confidentiality. If the message length is not a multiple of the block size of the underlying cipher in CBC mode, then message is padded suitably to make it of the right length. Although CTR mode does not necessarily require message padding but if the sender wishes to hide exact message length from attackers, then message padding can be used even in this mode.

Chen et. al at IEEE SP (2010) described techniques based on different packet sizes generated as various events take place in web applications to infer the state of the web-application. This attack could allow an attacker to breach the privacy of the user. At PETS 2012, Liu. et. al. proposed a scheme to pad messages in a group to make all the packets of the same size to achieve k-indistinguishability. They claimed that this scheme could withstand the attacks described in Chen et. al's work.

In this work, we analyze privacy and security aspects of encryption modes, padding schemes and order of padding of messages in TLS during encrypted communication between client and web-application on the server. We show that using padding schemes to pad all packets to hide message sizes during communication without considering underlying encryption modes and padding methodology is not safe .

We consider the technique of Liu et. al when certain combinations of encryption modes and padding schemes are used in TLS. We show that k-indistinguishability of packets does not always hold. In particular, we describe a chosen ciphertext attack to show that the MAC-PAD-ENCRYPT model to generate ciphertext in the TLS record protocol helps the attacker in disrupting privacy of traffic under certain conditions. We also show how a similar attack can be carried out on CCMP protocol used in WPA2 to maintain confidentiality and integrity in wireless networks if MAC-PAD-ENCRYPT is followed.

Acknowledgments

I would like to thank my parents for supporting me throughout my journey. Without their love and support this would not have been possible.

Firstly, I would like to thank Dr. Donghoon Chang who always motivated me to think harder and research deeply into any topic. I have seen very few highly motivated and hardworking people like Dr. Donghoon. From him I could learn how to think in multidimensional ways about any problem.

I sincerely thank Dr. Somitra Sanadhya to help me during my thesis work. He motivated me to take cryptography as a thesis subject in the first place. Dr. Somitra's effort towards perfection has made this thesis possible.

Thanks to Dr. Pankaj Jalote and other associated members for creating an institute like IIIT-Delhi. I feel really fortunate to be a part of this institute.

I am also lucky to have friends who have been there throughout my college life. At IIIT-Delhi also, I got accompany of some great friends like nishant, rohit(jain) , rohit(romley) , aritra, prateek, veeru, gajendra, anuj, ganesh, pankaj, noufal to name a few.

Lastly, I would like to thank Dr. Shweta Agrawal(IIT-Delhi), Dr. Debajyoti Bera(IIIT-Delhi) for their presence in thesis committee.

Sandipan Biswas
MT12018
IIIT-Delhi

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Observable attributes from encrypted network traffic, such as packet sizes, timing, etc., have been used to compromise the security of applications [2, 4, 11]. Even though encryption is used to ensure that confidentiality of the content of a packet is preserved, attributes like size of the packet, time taken between receiving the request and reply etc. can still be eavesdropped easily by an attacker. Attacks utilizing such techniques are known as side channel attacks. Side channel attacks on many web-applications were shown in [3] where the authors showed that privacy of AJAX based real-time web applications can be compromised by analyzing traffic attributes.

If messages are not padded to the same size then an attacker can guess the state of application of a user by observing the packet sizes. To mitigate traffic distinguishability based on packet sizes, two message padding schemes were proposed in [3]. These were named random padding and round padding. However, it was shown in [3] that such mitigation is costly due to extra overhead because of the padding. As a result, it was recommended in [3] to use an application specific padding instead of application agnostic padding. In similar direction, an approach known as k-indistinguishability was proposed in [5] which relies on an application agnostic padding scheme. This scheme works by grouping all packets which could be sent in response to a query. The reply to this query is formed with a packet padded in such a way that its size is the same as the maximum size of a packet in the group. This scheme provides k-indistinguishability if the group size is k.

TLS protocol [7] is used to ensure confidentiality and integrity of traffic at application level. TLS is a protocol suite which has a number of component protocols. In this work, we concentrate on the TLS record protocol which deals with symmetric key cryptography (block ciphers, stream ciphers and MAC algorithms) to ensure that a secure channel is established between a client and a server. Other protocols in TLS include TLS Handshake Protocol which is responsible for authentication, session key establishment and ciphersuite negotiation, and TLS Alert Protocol which carries error messages and helps traffic management. TLS record protocol uses MAC-PAD-ENCRYPT approach. In this approach, the MAC value is generated on application data excluding padding. Any padding, if required, is appended later. Thus, if the padding is used then it is not authenticated. During encryption, this data (including the padding) is encrypted

to generate the ciphertext using the selected cipher and encryption mode. It is also possible to use an authenticated encryption (AE) scheme combining both the authentication as well as encryption. However, use of AE schemes is not so common in current scenario [1].

Chosen ciphertext attacks like Lucky13 [1] have been demonstrated earlier where MEE construction in TLS record protocol was exploited. CBC encryption mode along with PKCS7 padding was used in the Lucky13 attack. This attack used time to compute MAC as the side channel measure. This attack reveals plaintext only when the attacker is on the same LAN because timing differences which are used to distinguish packets vary in the order of microseconds. Due to the small timing differences, this side channel attack over internet is not feasible.

However, the implementation aspects of the padding scheme of TLS in case of certain mode like counter mode etc. have not been considered in any previous work. In this work, we analyze the effect of padding scheme in conjunction with the mode of operation of the block cipher in TLS record protocol to attack the privacy of encrypted traffic.

Our Contribution We are specifically interested in the k-indistinguishability approach proposed by Liu et. al. [5] to mitigate the attack of [3]. We analyze the effect of an encryption mode along with a padding scheme on the k-indistinguishability. We consider CTR (counter) and CBC (cipher block chaining) modes and consider the bit padding scheme of the form 10*. While analyzing the scheme, we also consider the MAC-PAD-ENCRYPT construction in the TLS record protocol. We exploit unauthenticated padding added to messages during the encryption in TLS to demonstrate our attack. We propose truncation based chosen ciphertext attack on TLS record protocol to distinguish between packets in the same group having same sizes based on the error messages generated when wrong MAC is extracted during decryption. We consider the order of padding in TLS and discuss it's consequences in our attack. Our attack demonstrates that padding of messages should be done before MAC generation, and not after MAC generation in TLS as it can lead to privacy breach of traffic.

We also explore our attack principle on CCMP used in WPA2. Similar kind of truncation based attack can also be carried out on WPA2.

Chapter 2

Definitions

2.1 Block Cipher

In cryptography, a block cipher is a symmetric key cipher operating on fixed length groups of bits, called blocks, with an unvarying transformation. A block cipher encryption algorithm might take (for example) a 128-bit block of plaintext as input, and output a corresponding 128-bit block of ciphertext. The exact transformation is controlled using a second input, called the secret key. Decryption is similar. The decryption algorithm takes, in this example, a 128-bit block of ciphertext together with the secret key, and yields the original 128-bit block of plaintext. If the plaintext is not a multiple of block size then it is padded according to a standard scheme. A commonly used padding scheme is the "10*" padding. This scheme adds a 1 bit followed by sufficient number of 0 bits to the plaintext to make it a multiple of block length. The padding is unique and easily reversible. One only needs to remove all the bits up to the last 1 bit in the final block to get the unpadded message. The block cipher is typically denoted by the following notation.

$$E : \{0,1\}^k \times \{0,1\}^n \Rightarrow \{0,1\}^n$$

This notation means that E takes two inputs, one being a k-bit string and the other an n-bit string, and returns an n-bit string. The first input is the key. The second might be called the plaintext, and the output might be called a ciphertext. The key-length k and the block-length n are parameters associated with the block cipher. They vary from block cipher to block cipher, as of course does the design of the algorithm itself.

2.2 Side Channel Attack

Side Channel Attacks are attacks on software/hardware systems which utilize some weaknesses in the implementation of the cryptographic schemes rather than the design of the scheme itself.

Padding Oracle Oracle is a system which perform cryptographic operations on the request of a user (in our case attacker). A padding oracle is a specific type of oracle that will take encrypted data from the user, attempt to decrypt it privately, then reveal whether or not the padding is correct. We'll get into what padding is and why it matters soon enough. Padding oracle is realizable in practice in many vulnerable web-applications which upon receiving an encrypted text send back response informing whether the plaintext padding is valid or not.

2.3 Cipher-block chaining (CBC)

Let, P be the plaintext, P_n be the plaintext of length n blocks, $C =$ the corresponding ciphertext, $C_n =$ the ciphertext of length n blocks, $n =$ the number of blocks (P and C have the same number of blocks by definition), $IV =$ the initialization vector which is a random string frequently set to all zeroes, $E() =$ a single-block encryption operation (any block encryption algorithm, such as AES or DES, it doesn't matter which), with some unique and unknown (to the attacker) secret key (that we don't notate here), $D() =$ the corresponding decryption operation. We can then define the encrypted ciphertext C in terms of the encryption algorithm, the plaintext, and the initialization vector as follows:

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

$$C_n = E(P_n \oplus C_n), \quad \forall\ n > 1$$

CBC decryption is as follows:

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

$$P_n = D(C_n) \oplus C_{n-1}, \quad \forall\ n > 1$$

First block of ciphertext is decrypted, then XORed with the IV. Remaining blocks are decrypted then XOR'ed with the previous block of ciphertext. Note that this operation is between a ciphertext block that we control and a plaintext block that were interested in. This is important. Once all blocks are decrypted, the padding on the last block is validated. Here, the padding oracle checks if the padding is valid or not. Response from the oracle to the attacker reveals information which may lead to unauthorized decryption of the ciphertext. Ideally, a padding oracle should not be availale to the attacker i.e. an application should not inform a user about bad padding because this can leak information [9].

2.4 Counter Mode Encryption (CTR)

The Counter (CTR) mode is a confidentiality mode that features the application of the forward cipher to a set of input blocks, called counters, to produce a sequence of output blocks that are

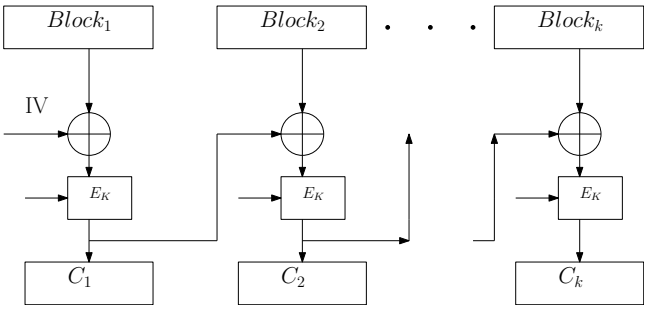


Figure 2.1: Cipher Block Chaining

exclusive-OR'ed with the plaintext to produce the ciphertext, and vice versa. The sequence of counters must have the property that each block in the sequence is different from every other block. This condition is not restricted to a single message: across all of the messages that are encrypted under the given key, all of the counters must be distinct.

In this recommendation, the counters for a given message are denoted $T_1, T_2, .., T_n$.

Given a sequence of counters, $T_1, T_2, \ldots, T_n$, the CTR mode is defined as follows:

CTR Encryption:

$$O_j = E_k(T_j) \quad for \quad j = 1, 2, \ldots, n$$

$$C_j = P_j \oplus O_j \quad for \quad j = 1, 2, \ldots, n-1$$

$$C_n^* = P_n^* \oplus MSB_u(O_n)$$

CTR Decryption:

$$O_j = E_k(T_j) \quad for \quad j = 1, 2, \ldots, n$$

$$P_j = C_j \oplus O_j \quad for \quad j = 1, 2, \ldots, n-1$$

$$P_n^* = C_n^* \oplus MSB_u(O_n)$$

where k is key associated with underlying block cipher.

In CTR encryption, the forward cipher function is invoked on each counter block, and the resulting output blocks are exclusive-OR'ed with the corresponding plaintext blocks to produce the ciphertext blocks. For the last block, which may be a partial block of u bits, the most significant u bits of the last output block are used for the exclusive-OR operation; the remaining b-u bits of the last output block are discarded.

In CTR decryption, the forward cipher function is invoked on each counter block, and the resulting output blocks are exclusive-OR'ed with the corresponding ciphertext blocks to recover the plaintext blocks. For the last block, which may be a partial block of u bits, the most significant u bits of the last output block are used for the exclusive-OR operation; the remaining b-u bits of the last output block are discarded.

In both CTR encryption and CTR decryption, the forward cipher functions can be performed

in parallel; similarly, the plaintext block that corresponds to any particular ciphertext block can be recovered independently from the other plaintext blocks if the corresponding counter block can be determined. Moreover, the forward cipher functions can be applied to the counters prior to the availability of the plaintext or ciphertext data.

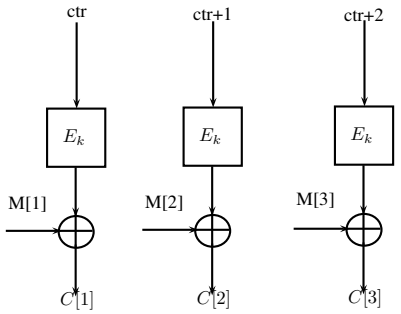


Figure 2.2: Counter Mode Encryption

2.5 Padding

Bit-Padding For a plaintext block M and block cipher or Hash function of minimum input size $|X|$, If $|M| < |X|$, the message M is padded with 1 followed by 0's where Padded message $= M|10^(|X| - |M|)$.

Bit padding can be applied to messages of any size. A single set ('1') bit is added to the message and then as many reset ('0') bits as required (possibly none) are added. The number of reset ('0') bits added will depend on the block boundary to which the message needs to be extended. In bit terms this is "1000...0000". This method can be used to pad messages which are any number of bits long, not necessarily a whole number of bytes long. For example, a message of 23 bits that is padded with 9 bits in order to obtain a 32-bit block is as shown in Figure 2.3 below.

10110111100111111100110000110	100000000

Figure 2.3: Bit-Padding

2.6 Message Authentication Code(MAC)

MAC ensures authenticity and integrity of messages sent between two communicating parties. Cryptographic hash function can be used to generate MAC. Essentially, the message authentication code (MAC) is a small xed-size block of data that is generated based on a message M of variable length using secret key K as follows. It is also called cryptographic checksum.

$$MAC = C(K, M).$$

If A wishes to send B a message M, and protects it via a MAC, they rst need to share a secret key K for the authentication (in addition to an encryption key if they also use encryption). Then A calculates code MAC as a function of M and K. Then the message M plus the code MAC are transmitted to B. B performs the same calculation on M, using K to generate a new code MAC. The received code MAC is compared to the calculated code MAC to verify the data integrity. As only A and B have the secret key and hence are able to generate the correct MAC, source authentication is also achieved.

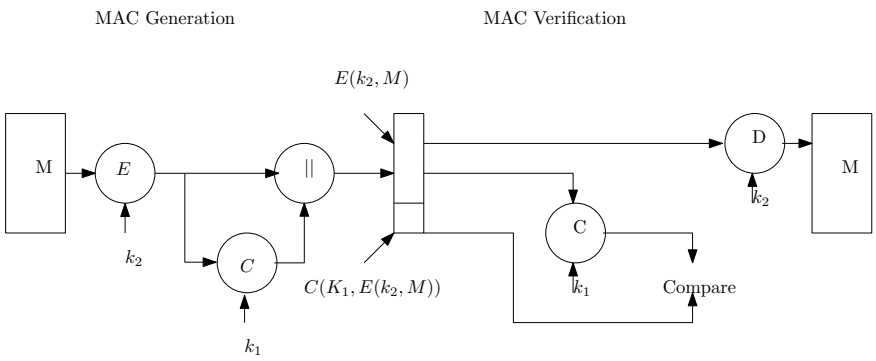


Figure 2.4: Message Authentication and confidentiality.Authentication tied to ciphertext.

2.7 TLS Protocol

The TLS protocol provides communications security over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery. According to the protocol's creators, the goals of the TLS protocol are cryptographic security, interoperability, extensibility, and relative efficiency.

These goals are achieved through implementation of the TLS protocol on two levels: the TLS Record protocol and the TLS Handshake protocol.

TLS Record Protocol The TLS Record protocol negotiates a private, reliable connection between the client and the server. Though the Record protocol can be used without encryption, it uses symmetric cryptography keys, to ensure a private connection. This connection is secured through the use of keyed hash functions by using a Message Authentication Code.

TLS Handshake Protocol The TLS Handshake protocol allows authenticated communication to commence between the server and the client. This protocol allows the client and the server to speak the same language, allowing them to agree upon an encryption algorithm and encryption keys before the selected application protocol begins to send data.

Chapter 3

Attack model and Previous work

Before explaining previous work and our attack in next chapters, we describe our attack model. Then we also explain related previous work.

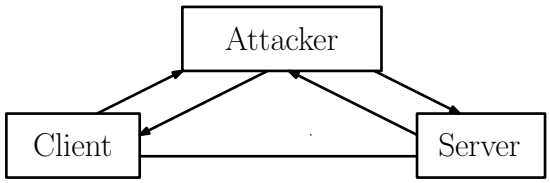


Figure 3.1: Attack Model

In this attack model we have three components: Client, Server, Attacker. Client and Server both are assumed to be secure. Client encrypts a plaintext before sending it to the server and the server decrypts it based on the shared secret key. The attacker is able to either passively monitor network traffic or actively modify network traffic contents. Consider a scenario where the client is accessing a search engine over a secure HTTPS connection and the search engine is AJAX enabled. As soon as the client starts typing search keywords, application requests are generated. Now Attacker monitoring the channel can sniff the packets generated due to requests. We also consider the scenario where the attacker can modify the packets and resend the request to server. In our attack we consider client and server in a peer-to-peer network where both client and server can give inputs to each other.

3.1 Previous Work

Attacks on TLS/SSL have been studied extensively over the last few years. There have been attacks designed on various components of TLS/SSL protocol. For example, the design and weaknesses of cipher suites used in TLS encryption and key establishment have been studied a lot in the past. BEAST [6] attack by Doung et. al. showed how a chosen plaintext attack can be mounted by an attacker exploiting IV which is same as the previous ciphertext block used in CBC mode of TLS. This attack idea works for long messages. Note that since ciphertext can

be eavesdropped, attacker will have to just guess the plaintext corresponding to the ciphertext to be decrypted and repeatedly send the guessed plaintext to an encryption oracle and match. Another attack known as CRIME [8] by Rizzo et. al showed how session tokens or other secret information can be discovered based on compressed size of HTTP requests. CRIME is a brute-force attack that works by leveraging a property of compression function, and noting how the length of the compressed data changes. Even though TLS encrypts the content in the TLS layer, an attacker can see the length of the encrypted request passing over the wire, and this length directly depends on the plaintext data which is being compressed. Finally, an attacker can make the client generate compressed requests that contain attacker-controlled data in the same stream with secret data. The CRIME attack exploits these properties of browser-based SSL. An extension of this attack known as TIME was proposed by Be'ery et. al which is a choosen plaintext attack on HTTP responses. The attack model of CRIME gives information about plaintext based on the length of encrypted and compressed data. TIME uses this model and timing information differential analysis to infer the compressed payload's size. BREACH [10] by Gluck et. al revived the CRIME attack by targeting the size of compressed HTTP responses and extracting secrets hidden in the response body. As a result of all these attacks, the compression is now disabled by default in TLS.

We focus on an attack known as Lucky13 [1] showed by AlFardan and Paterson which is a chosen ciphertext attack based on subtle timing differences caused during decryption. The attack allows a man-in-the-middle attacker to recover plaintext from a TLS connection when CBC-mode (cipher-block chaining) encryption is used. There is a subtle timing bug in the way that TLS data decryption works when using the (standard) CBC-mode ciphersuite. Lucky Thirteen uses the same attack mechanism as the padding oracle attack [9].

Side channel attacks on web applications were formally researched by Chen et. al [3]. The authors modeled a web application formally and introduced parameters to decide if a web-application is vulnerable to such attack. AJAX based web-applications generate packets when an event takes place in the application. It was shown that based on the sizes of packets generated during an event, a passive attacker can guess the state of application. As a mitigation approach they experimented with application-independent padding like random padding for packets or round padding which incurred significant overhead. Hence application specific padding was suggested to mitigate such attacks.

Further working on padding schemes, Liu. et. al [5] took two factors into consideration while providing mitigation to this attack: (1) The Cost of padding should be minimized to be practically used, and (2) The privacy requirement that packets are indistinguishable from each other is satisfied. The authors introduced the concept of k-indistinguishability which states that packets should be grouped into k sized groups such that packets in same group are of the same size. They proposed novel algorithms to group inputs such that corresponding packet sizes are identical to each other. As a result, it was shown that while privacy of traffic is maintained ,efficiency has increased and overhead has decreased. This suggests that such approaches are effective.

Chapter 4

Effect of padding order in TLS on k-indistinguishability of traffic

In this chapter, we give an idea of how to break k-indistinguishability of a padded set of packets using chosen ciphertext attack given an Oracle which determines whether MAC is valid or invalid. Each user action in a web-application generates traffic which in turn generates a response from the server. For example, in a search engine when a user searches a query then each keystroke generates a list of suggestions which has a specific size. This suggestion list size varies as more characters are typed by the user. An attacker can use observed sizes and this allows him to map the lists to the user inputs. Now, when an attacker monitors network traffic generated from the use of a search engine, based on packet size of response, he can get information about user inputs. To prevent such distinguishability of packets, user actions are grouped in such a way that uniform traffic is generated whenever an input from that group is sent to server.

Packets corresponding to actions are padded in the same group if both actions have action as a prefix. For example, user types {**c,d**} and {**c,e**} as two different action sequences in a typical search engine. Now if grouping is done such that {**d,e**} is grouped in the same group as prefix {**c**} then the attacker cannot distinguish between two different action sequences since {**d,e**} will be padded to the same size as the other action sequence.

After such padding, packets in the same group are indistinguishable from each other. Our attack shows that this claim of k-indistinguishability of packets leading to privacy of user-actions is incomplete without careful consideration of underlying encryption modes along with position of padding during MAC generation and encryption of packets. Various encryption modes like Counter Mode, Cipher Block Chaining etc. have different effects on privacy when used in such cases. We also show how MAC-PAD-ENCRYPT is not safe in counter mode as well as Cipher Block Chaining mode. In our attack we show how usage of counter mode and CBC mode with specific padding scheme like Bit-padding is not privacy safe under chosen ciphertext attack.

We propose a truncation based attack which can disrupt privacy of k-indistinguishability of traffic by exploiting underlying padding scheme and encryption modes in case of MAC-PAD-

ENCRYPT scheme in TLS. Truncation based attack is explained both on cipher block chaining as well as counter mode. In counter mode, the truncation is bit based and in case of cipher block chaining mode, the truncation is block based.

Our Assumptions

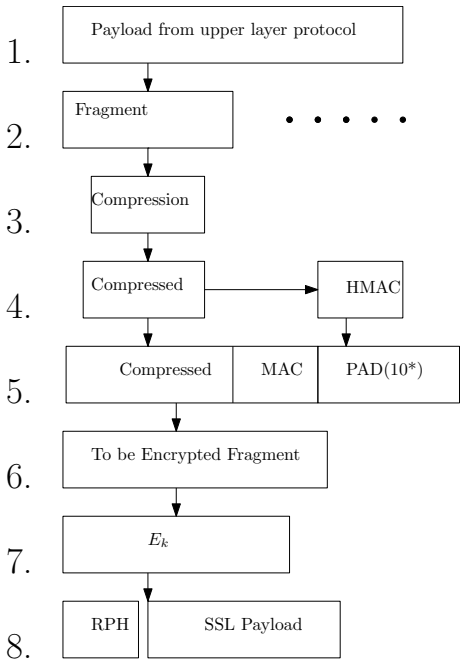
- We assume that k-indistinguishability is already implemented at server for that particular application for all possible action sequences.
- To implement padding for k-indistinguishability, bit-padding (10*) has been used.
- Attacker can modify ciphertexts and send it to server for decryption.
- Attacker can monitor network traffic generated due to use of the web-application.
- Messages have been padded after MAC is computed. Since padding is not necessary in Counter mode in general, padding after MAC computation is a possibility. Further, padding before MAC computation on the messages can be a overhead.(This assumption is valid since in TLS and DTLS both padding is done after MAC is computed.This construction has led to earlier attacks as shown in [9].)
- Our work does not require strict client-server interaction. Both the communicating parties can perform actions and generate responses to each other, as in a peer-to-peer system.

Theory of the Attack

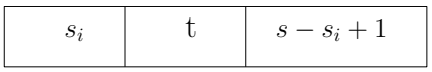
Let us consider that we have j number of packets and respective sizes in group are $G = [s_1, s_2, \ldots, s_j]$, where s_i is size of a packet, after applying k-indistinguishability on packets generated while client was interacting with server. All packets belonging to the same group must be padded to the maximum size of these packets to make these packets indistinguishable from each other within the same group. Before padding, the MAC will be calculated on all packets to ensure integrity. Assume that the size of the MAC is t. The MAC is appended to each packet in the group. After MAC computations, the packets are $S = [s_1 + t, s_2 + t, \ldots, s_j + t]$ where $j \geq k$. Bit-padding is done at this stage and after this, the packet sizes are:

$$S = [s_1 + t + (s - s_1 + 1), s_2 + t + (s - s_2 + 1), \ldots, s_k + t + (s - s_j + 1)],$$

where s is the maximum size of a packet in a group and 10^{s-s_i+1} is the padding for i^{th} packet, i.e. $s - s_i + 1$ is the size of padding to make all packets in the same group equal in size. Note that in counter mode, if the message size is greater than block size of the underlying cipher, then the message is broken into blocks. However, the last block doesn't have to be of the block size of the underlying block cipher. This is unlike the behavior in case of CBC mode. After padding is done, encryption takes place in the mode being used (counter mode or cipher block chaining mode).



(a) Processing a message by Record Protocol in TLS. Here RPH denotes record protocol header.



(b) MAC generation(t) and subsequent padding

Figure 4.1: (a) Encryption of a message in TLS protocol (b) Size of a message after MAC generation and padding i.e. s_i = Actual plaintext size, t= Size of MAC tag, $s - s_i + 1$ = Size of padding (10*) added.

We consider TLS protocol in this attack scenario. In Figure 4.1, we assume padding is done at stage 5, i.e. after MAC is generated and padding is done using Bit-Padding. Our attack analyzes the effects when such scenario takes place in the context of k-indistinguishability scheme proposed by Liu et. al. [5].

Effect Of Counter mode In counter mode encryption, we can use bit based truncation where attacker can truncate number of bits from each message to distinguish a packet from others in same group.

To learn which packet corresponds to which input, the attacker can follow truncation based approach. The attacker considers packet for which padding size is maximum. Let's assume the maximum padding size to be d. The attacker truncates $d-1$ bits from all packets and sends the modified packets to the server. This modification may corrupt all other packets (Case 2 below) but the current packet in consideration (Case 1 below). Thus the attacker can distinguish current packet from others. Note that the number of bits interpreted as padding depends on the position of first 1 from the LSB which is denoted in Figure 4.1 as l. This could be inside the computed MAC value, or the message itself. In either of the cases, the actual encrypted part of the packet will be corrupted (since a part of it will be truncated).

For i^{th} packet after truncation and decryption at server, there are two cases as follows.

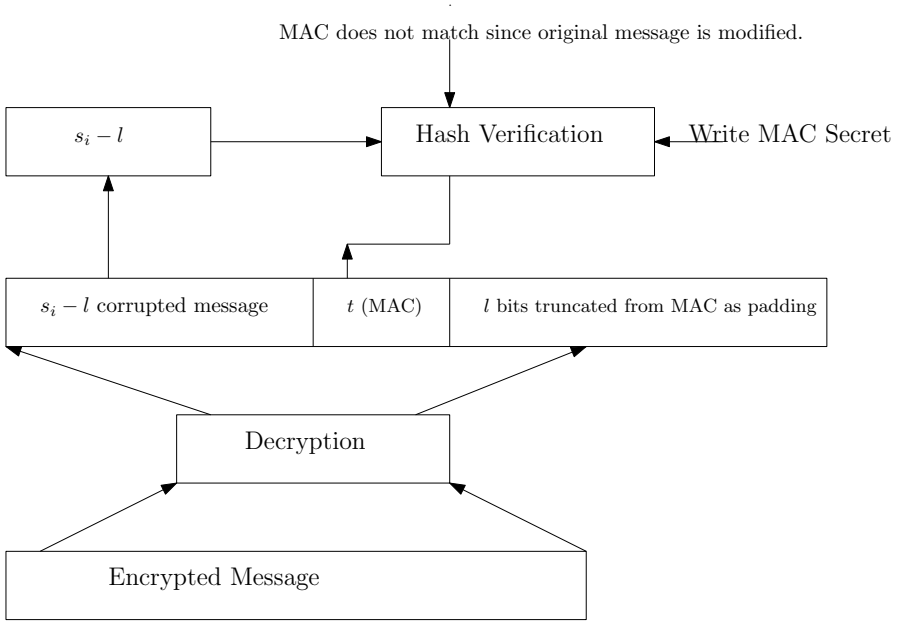


Figure 4.2: MAC is corrupted due to truncation of original MAC. In the figure l denotes no of bits truncated from MAC and interpreted as padding.

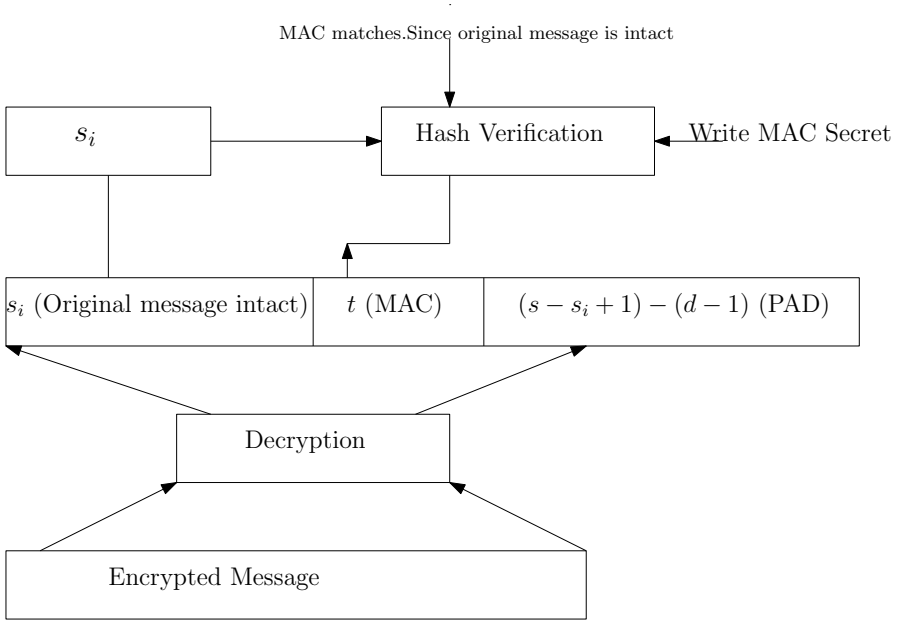


Figure 4.3: MAC is not corrupted.

- If $(s - s_i + 1) \geq (d - 1)$ then MAC will not be corrupted and will be valid for message after padding removal Figure 4.3.
- If $(s - s_i + 1) < (d - 1)$ then a part of MAC will be stripped off as padding after decryption, generating wrong MAC, some part of which will be from the message itself. Packets which correspond to this case generate invalid MAC error which helps attacker in distinguishing a particular packet from others in same group Figure4.2.

To distinguish other packets, the attacker has a smaller group and hence much better probability of distinguishing a packet. Same procedure as described above is followed for the remaining groups. In this way, an attacker can break k-indistinguishability of traffic using such truncation based attack.

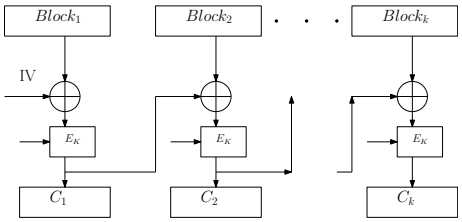
Effect Of Cipher Block Chaining Mode In cipher block chaining mode we cannot use bit truncation based attack since underlying block cipher needs input to be multiples of block size. Hence we propose block based truncation instead of bit based truncation. However padding size of packets needs to be greater than the block size of the underlying block cipher. Only then a possibility exists that such block based truncation may work.

Here size of maximum in a group needs to be padded to multiple of block size. Thus all other packets in same group needs to be padded to the padded size of maximum packet size in group.

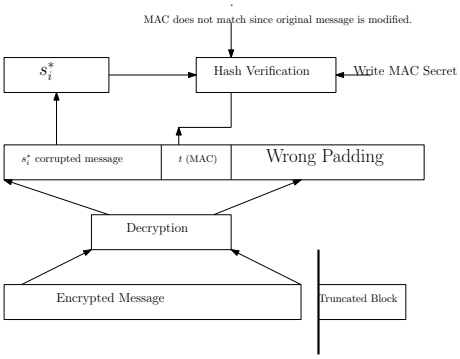
Let's say size of largest packet in a group is s_{max} which is not a multiple of underlying block cipher input size n. Hence packet needs to be padded according to bit-padding with $10^{n-s_{max}}$ with a resulting size of $k * n$ where $k > 0$ is a non-negative integer such that $k * n \geq s_{max}$. Resultant size of all packets in same group will be

$$S = [s_1 + t + ((k * n) - s_1), s_2 + t + ((k * n) - s_2), \ldots, s_k + t + ((k * n) - s_k)].$$

All these packets will be divided into b blocks of input size of encryption scheme n used in the CBC mode.



(a) Message is divided into blocks of size n,where n is input size of block cipher before encryption at client side



(b) If truncated block contains padding including 1 from padding,server generates invalid MAC error at server

Figure 4.4: (a) Attacker removes k^{th} block from each message such that it generates error for intended message only. (b) MAC is corrupted due to block truncation of original MAC.

The attacker removes last block from each message such that MAC generated at server turns out to be invalid resulting in an error message. Thus the attacker can distinguish one message from others based on block based truncation scheme. After distinguishing one block, the attacker can follow the same step for others and hence break k-indistinguishability.

4.0.1 Example based on CTR mode

We will explain our attack based on mitigation scheme proposed by Liu et. al. [5]. We build an example first based on proposed scheme and explain our attack on the example.

Example based on k-indistinguishability mitigation scheme

Before we explain example based on k-indistinguishability, first let's define some terms.

Given a Web application, we define

- an action **a** as an atomic user input that triggers traffic, such as a keystroke or a mouse click.
- an action-sequence $a^{\rightarrow}$ as a sequence of actions with known relationships, such as consecutive keystrokes entered into real-time search engine or a series of mouse clicks on hierarchical menu items. We use **a[i]** to denote the i^{th} action in **a**.
- an action-set A_i as the collection of all the i^{th} actions in a set of action-sequences. We will simply use A if all action-sequences are of length one.
- a flow-vector **v** as a sequence of flows where each flow **s** is an integer (a directional packet size). An action corresponds to a flow-vector based on packets it triggers.
- a vector-sequence $v^{\rightarrow}$ as a sequence of flow-vectors corresponding to an equal-length action-sequence $a^{\rightarrow}$, with each **v[i]** corresponding to **a[i]**.
- a vector-set V_i (or simply $\mathbf{V}$) as the collection of all the i^{th} flow-vectors in a set of vector-sequences, which corresponds to an action-set in the straightforward way.
- SVSD (Single Vector,Single Dimension) can be used to model vector action set for traffic where there is only one action, each action has only one flow in corresponding flow vector. Here single vector corresponds to single action in an action sequence. Single dimension refers to single packet flow generated when an action takes place.
- SVMD (Single Vector,Multiple Dimension) can be used to model vector action set for traffic where there is only one action, each action has only multiple flow in corresponding flow vector. However for all actions in an action set have equal number of flows in a flow vector. Thus dummy packets need to be inserted in case some actions have unequal number of flows. Here single vector corresponds to single action in an action sequence. Finally, multiple dimension refers to multiple packet flows generated due to an action.
- In MVMD (Multiple Vector,Multiple Dimension), an action in an action-set generates multiple flows where multiple action are there in action sequence.

Example 1: Assume a and 00 to be the only possible inputs, there are two action-sequences **a** and **00**, and two action-sets $A_1 = \{a,0\}$ and $A_2 = \{0\}$.

Example 2: Following Example 1, we have three flow-vectors, $v_1 = 509$, $v_2 = 505$, and $v_3 = 507$ (note that we only model those packets whose sizes can help to identify an action), corresponding to actions a, 0 (as first keystroke), and 0 (as second keystroke), respectively. We have two vector-sequences, v_1 and v_2v_3, corresponding to action-sequences a and 00, respectively. We also have two vector-sets $V_1 = \{509, 505\}$ and $V_2 = \{507\}$ corresponding to the two action-sets A_1 and A_2 in Example 1.

We assume 4 action sequences all of equal length. For example, $a_1 = \{a, b\}$, $a_2 = \{b, c\}$, $a_3 = \{c, a\}$ and $a_4 = \{a, d\}$.

Note that: a_1 and a_4 have same prefix for second keystroke. Prefix is $\{a\}$.

Corresponding vector sequences are : $v_1 = \{509, 487\}$, $v_2 = \{504, 482\}$, $v_3 = \{502, 501\}$, $v_4 = \{509, 497\}$.

Note: 509 is for single keystroke only$\{a\}$. But 487 is for $\{a, b\}$,two consecutive keystrokes. Size of bytes transferred for consecutive keystrokes may be smaller or larger than single keystroke. It depends on the application.

Now according to [5], the vector-set can be formed as $V_1 = \{509, 504, 502, 509\}$, and $V_2 = \{487, 482, 501, 497\}$.

Similarly Action-Set: $A_1 = \{a, b, c, a\}$ and $A_2 = \{b, c, a, d\}$.

There will be two Vector-Action Set : $VA_1 = \{V_1, A_1\}$, $VA_2 = \{V_2, A_2\}$. They will come out to be:

$VA_1 = \{(a, 509), (b, 504), (c, 502), (a, 509),\}$, and

$VA_2 = \{(b, 487), (c, 482), (a, 501), (d, 497)\}$.

Input	Original Packet Size
a	509
b	511
c	508

Table 4.1: Single Vector,Single Dimension

Input	Original Packet Size	Prefix
b	487	a
c	482	b
a	501	c
d	497	a

Table 4.2: Single Vector, Multiple Dimension based on prefix based padding.

After grouping, applying Simple SVSD (Single Vector,Single Dimension) algorithm on first table : $SVA_1 = \{(c, 508), (b, 511), (a, 509)\}$, $PVA_1 = \{(c, 511), (b, 511), (a, 511)\}$[Padding].

After applying SVMD (Single Vector,Multiple Dimension) algorithm and padding: $SVA_2 = \{(c, 482), (b, 487), (d, 497), (a, 501)\}$, $PVA_2 = \{(c, 501), (a, 501), (b, 501)(d, 501)\}$.

For same input string two flows corresponding to ab and ad $\{511, 501\}$,$\{511, 501\}$.

Now attacker cannot distinguish between two input strings. Hence it maintains k-indistinguishability.

Attack on the example based on k-indistinguishability

From the above example, we have a group in which all the packets have same sizes after padding. Now, lets explain our attack on this group to distinguish between inputs present in the set. We explain our attack in a scenario where bit-padding and encryption modes like CBC and CTR are used.

To pad packets that belong to the same set of size k, there are two options.

- Pad message in step 3 or 4 (refer Figure 4.1). This scenario is PAD-MAC-ENCRYPT.
- Pad message after MAC is calculated i.e. in step 6(refer Figure 4.1). This is MAC-PAD-ENCRYPT.

In our attack we show why MAC-PAD-ENCRYPT is unsafe if k-indistinguishability is used to maintain traffic anonymity. In the following example, we use MAC-PAD-ENCRYPT to demonstrate that such attacks are possible. Our attack works because the padding is unauthenticated.

Lets say MAC computed on messages is of size 32 bit. Now each packet is padded to the maximum size of the packet in the set. This is known as Ceiling Padding. Bit-padding is used for the padding.

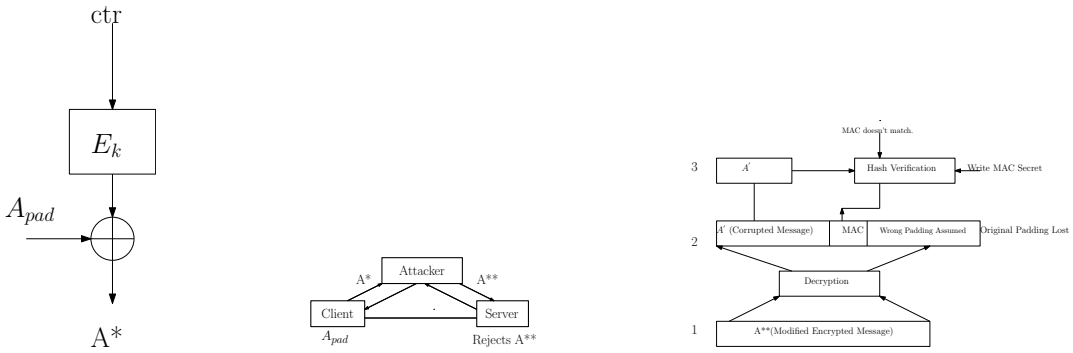
For our target example after bit-padding packets are padded according to following table:

Packet Padded Output	Input	Original Size	Pad
A_{pad}	**a**	509+32	100
B_{pad}	**b**	511+32	1
C_{pad}	**c**	508+32	1000

Table 4.3: Packets after MAC and padded according to bit-padding. Here MAC generated is appended to message first. Padding is done after MAC generation.

As mentioned earlier, our target example query set has {**a**,**b**,**c**} all of which has equal size. In our example attacker is assumed to modify all other members in same set at the same position such he can identify **c**. In the following step-by-step approach we have explained how attacker is able to do so.

Let's assume client types **a** first. After encryption of padded query A_{pad}, A* is captured by the attacker and he truncates 3 bits from all packets in same group. The modified packet is sent to the server. Server strips off padding from ciphertext after decryption. Now according to bit-padding, a part of message is stripped off as pad starting from LSB until 1 is encountered. Since padding is already removed by the attacker, part of a MAC is considered to be pad and hence wrong value of MAC is taken. The non-padded message is divided into MAC part which is used to verify the data part in the message. Since MAC part is changed it will be flagged as invalid and error generated. Thus attacker can know this packet is not the intended one i.e **c**. This attack process is described in Figure 4.5.



(a) During Encryption

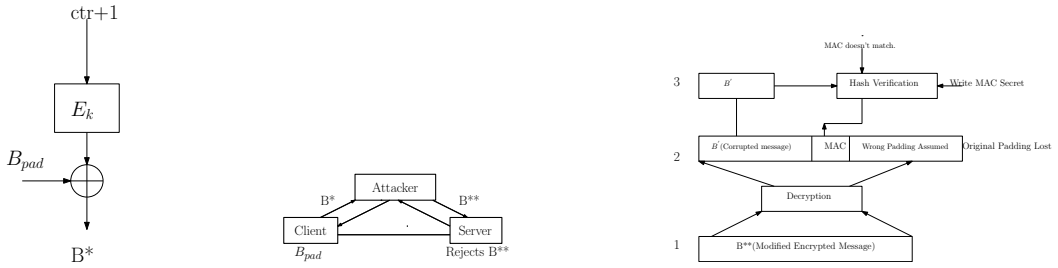
(b) Attacker modifies A*.Server respond with error message

(c) During decryption of modified ciphertext

Figure 4.5: (a) A_{pad} Encryption(Client Side) (b) Attacker truncates last 3 bit of ciphertext (c)Error generated during decryption due to wrong MAC

Modified Input	Encrypted Input	Original Packet Size	Packet size after truncation
A**	A*	509+32(MAC)+3(100)	$A^{'}$+32 +(Assumed Padding from MAC)

Table 4.4: Step 1. Attacker Modified packet content for A*. Part of MAC will be extracted as padding at server side according to Bit-padding. Thus wrong invalid MAC will generate error. Here $A^{'}$ is the resulting message size after truncation and padding is stripped off during decryption.



(a) During Encryption

(b) Attacker modifies B*. Server again respond with error message

(c) During decryption of modified ciphertext

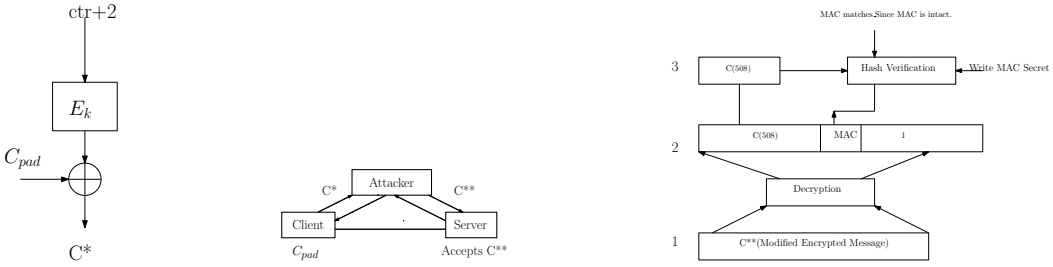
Figure 4.6: (a)B_{pad} Encryption(Client Side) (b)Attacker truncates last 3 bit of ciphertext (c)Error generated during decryption due to invalid MAC

Let us assume that the client types **b** as the query. After padding and encryption, B* is received by attacker, see Figure 4.6. The attacker truncates last 3 bits of packet (B**) and sends it to server to be decrypted. This modification corrupts original MAC part of the packet. The server strips off padding (as explained in case of A*) out of received packet after decryption. Since MAC part is changed it will be flagged as invalid and error is generated. Thus attacker can know this packet is not the intended one i.e **c**. This approach is described in Figure 4.6.

Modified Input	Encrypted Input	Original Packet Size	Packet size after truncation
B**	B*	511+32(MAC)+1(Pad)	$B^{'}$+32(MAC) +(Assumed Padding from MAC)

Table 4.5: Step 2.Attacker Modified packet content for B*. Part of MAC is removed as padding in B**.Invalid MAC generated. This causes MAC to be invalid at server. Here $B^{'}$ denotes resulting message size after truncation and padding is stripped off after decryption at server.

Let us assume that the client types **c** as the query. After padding and encryption C* is received by attacker.



(a) During Encryption

(b) Attacker modifies C*

(c) During decryption of modified ciphertext

Figure 4.7: (a)C_{pad} Encryption(Client Side) (b)Attacker truncates last 3 bit of ciphertext (c)C** data is not corrupted during decryption after pad removal. MAC remains intact. Server accepts.

Attacker next modifies ciphertext to identify **c** by truncating last 3 bits to create C* and sends it to the server. But this time server accepts the query since no modification in MAC has taken place even after padding is stripped off. MAC matches the actual content in packet and the server accepts. This is explained in Figure 4.7.

Modified Input	Encrypted Input	Original Packet Size	After truncation packet size
C**	C*	508+32(MAC)+4(1000)	$C^{'}$(508)+32(MAC)+1

Table 4.6: Step 3. Attacker Modified packet content for C*. Original MAC intact even after padding.Here $C^{'}$ denotes resulting message size after truncation and padding is stripped off after decryption at server. Note that $C^{'}$ is same as original message encrypted at client side.

Thus attacker identifies the plaintext query corresponding to a ciphertext. Thus k-indistinguishability is disrupted since attacker can distinguish one character from others in the group.

4.0.2 Example based on Cipher Block Chaining Mode

The attack described above also works when the underlying encryption mode is CBC. Whenever padding size in a set is more than block cipher size there is a chance that such attack may work. However since CBC mode works on blocks, the truncation is also required to be done based on blocks. Let us consider a scenario where we have 3 packets in a set which maintains 3-indistinguishability by padding each packet to maximum packet size in set. Further, we assume that underlying block cipher takes 128 bit input.

Set of packets size $S = \{257, 101, 129\}$ corresponding to inputs say $\{i_1, i_2, i_3\}$. Assume MAC is already included in packet sizes. Each packet is padded to maximum size in set S. Since input size of block cipher is smaller than amount of padding, packets has to be broken into chunks of 128 bit blocks.

During encryption packets are segregated into blocks as shown in Table 4.7.

Block1	Block2	Block3
128	128	1+127 ($10^{126}pad$)
101+27(10^{26})	128(0^{128})	128(0^{128})
128	1+127(10^{126})	128(0^{128})

Table 4.7: Message blocks size before encryption. Added numbers denotes size of pad added.In brackets actual pad contents added is shown.

Now if we apply our truncation based attack in this case, we can distinguish between packets. However here truncation will be block-wise unlike in counter mode where truncation was bit based.

To distinguish i_1 all that the attacker has to do is truncate the last block in all messages. This truncation will only affect the packet corresponding to i_1 since this will corrupt a part of message and after decryption at server, a part of the actual message will be misinterpreted as MAC according to bit padding removal method i.e. removing bits from last until 1 is encountered. This MAC extracted from message will be invalid due to truncation and will produce error message. Attacker can therefore distinguish which input corresponds to the modified ciphertext.

Similarly, to distinguish packet corresponding to i_3, the attacker can remove last two blocks in ciphertext corresponding to $\{i_2, i_3\}$. This truncation will lead to error message due to invalid MAC in case of i_3 but not i_2, which distinguishes i_3 too.

Thus we can see how privacy can be compromised using the encryption mode like CBC too using truncation based chosen ciphertext attack.

Chapter 5

Effect of padding order in WPA2 on k-indistinguishability of traffic

In this chapter, we discuss how the order of padding matters in encryption mode used in WPA2, a protocol used in wireless networks to maintain confidentiality and integrity affects privacy. In WPA2, AES-CCMP became a standard and was introduced in IEEE 802.11i. CCMP is an enhanced data cryptographic encapsulation mechanism designed for data condentiality and integrity and is based on the Counter Mode with CBC-MAC. Both the primitives use the AES standard. Vulnerability discovered earlier in WEP protocol had led to the introduction of WPA2 as the revised version of the wireless messaging protocol. AES-CCM is also one of the recommended cipher suite in TLS1.2.

5.1 CCMP processing in WPA2

CCMP or counter mode with cipher chaining mode was introduced in WPA2 to provide confidentiality as well as message integrity, in which the underlying block cipher is AES128. AES provides a method for encrypting data, obscuring the content so it cannot be read by an attacker. However, and just as important, the receiver needs to know that the message is authentic, i.e. it has not been modified. This is usually accomplished by adding a message integrity code (MIC). For efficiency, we want this MIC to be computed using the AES encryption algorithm so it makes sense that the operating mode should define how to provide both encryption and authenticity. This scheme is shown in Figure 5.1.

The encryption flow of a packet as shown in Figure 5.1 can be described as follows:

- First the MSDU packet that is coming from upper layer i.e. Logical Link Layer above MAC layer is fragmented into MPDU packets.
- Next the packets are processed using CCMP which uses counter mode for encryption and CBC-MAC for message authentication. For both of them, AES is used as an underlying

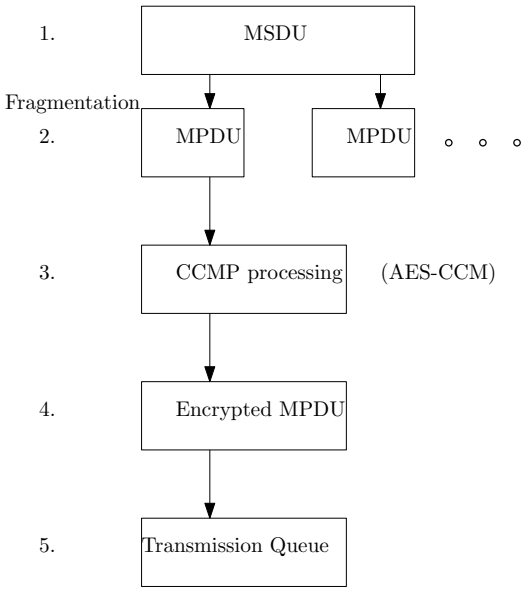


Figure 5.1: Fragmentation of MSDU from Logical Link layer into MPDU used in Medium Access Layer.And subsequent processing of MPDU in MAC layer.Note CCMP processing is done on MPDU not MSDU.

cipher. CCMP adds some header called "CP header" related to its protocol along with existing MAC header in MPDU.

- To protect message integrity, CBC-MAC is used to generate Message Integrity Code (MIC). To compute the MIC, the padding algorithm operates on the these three inputs: the MAC Header, the CP header and the padded plaintext data. The scheme uses AES as the underlying block cipher. The first block of message is generated using fields from the two headers. Subsequent blocks of messages are encrypted and XOR'ed with previous blocks according to CBC-MAC. Note that MAC header and CP header is only authenticated, but not encrypted.

- To maintain confidentiality, both the MIC and plaintext data are encrypted using AES in counter mode. No padding is required since the counter mode is used.

The counter mode does not hide plaintext packet sizes, since no padding is necessary in general in this mode. Therefore, it is necessary to pad the messages to achieve k-indistinguishability. In the case of AES-CCMP (used in WPA2, as mentioned earlier), padding can be done to hide plaintext sizes. Position of padding can be either before or after the MAC computation. We can similarly explore the effects of padding done after MAC computation as shown in step 5 of Figure 5.2. Note that we have considered unauthenticated padding here too. To distinguish packets padded according to k-indistinguishability we can similarly show a truncation based attack on AES-CCMP in WPA2 as we have shown in TLS.

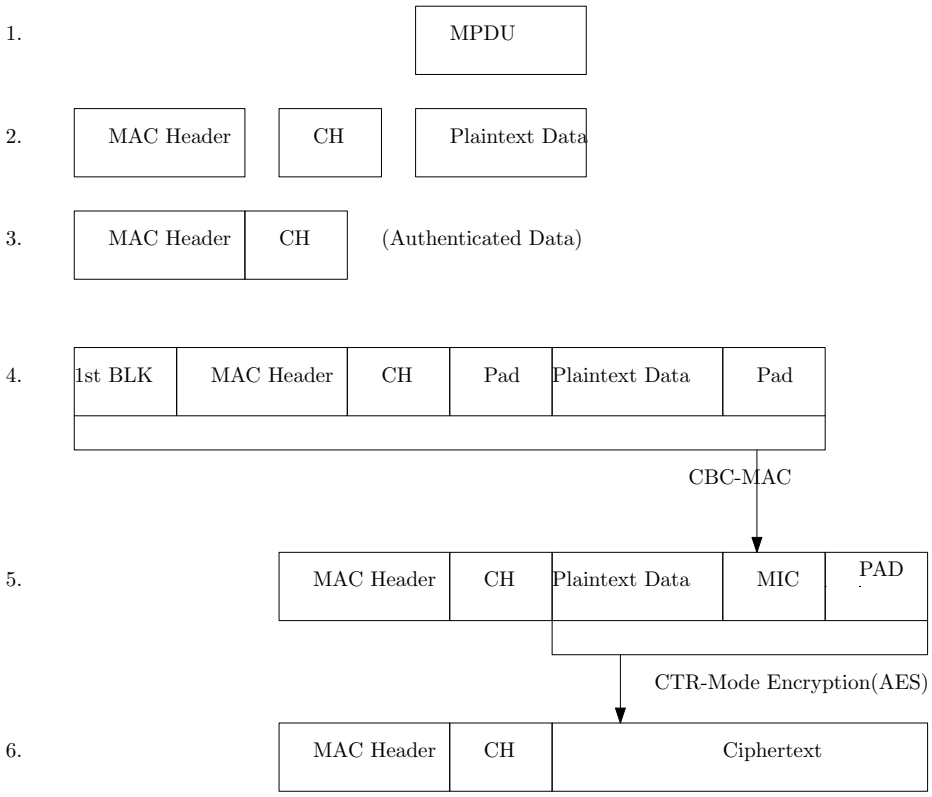


Figure 5.2: CCMP processing of a MPDU

5.1.1 Mitigations

In the attack explained earlier, we exploited the vulnerability caused due to the MAC-PAD-ENCRYPT model used in TLS. As we have explained in earlier sections, if MAC is computed and padding done after the MAC computation then such a scenario can lead to chosen ciphertext attacks of the kind described. If padding is applied before MAC computation then it would not have been possible to carry such an attack. If the padding type is bit-padding, then the size of the pad should be included in the header and subsequently included into MAC computation as well. This will allow the attacks of the type we described earlier as the padding length can be checked in the header by the legitimate use. PKCS7 padding can be used to pad, however this may be susceptible to padding oracle attacks as shown in [9].

5.2 Future Works

- We have not yet implemented the proposed attacks. To do so will require one to first implement Liu et. al's scheme [5]. Implementation of the earlier scheme and our attack is a possible line of work.
- Once can study other modes of operation which are vulnerable to the type of attack described in this report. Among authenticated encryption modes we have explored AES-CCM. Other possible vulnerable modes could be AES-GCM used in TLS, SSH etc.

- It is interesting to study other padding schemes and whether our attack can work against them.

5.3 Conclusions

Traffic indistinguishability is a hard problem. Liu et al. proposed an application agnostic approach (k-indistinguishability) to pad packets to same sizes. Our attack shows that indifference to underlying padding schemes and encryption modes and MAC calculation procedure can cause such privacy preserving schemes to break. Thus schemes to make traffic anonymous not only have be application agnostic and efficient, but they must also consider how the scheme affects the underlying cryptographic operation. A detailed analysis of padding scheme, mode of operation and position of padding are necessary to prevent side channel attacks like the one shown in this report.

Bibliography

[1] AlFardan, N. J., and Paterson, K. G. Lucky thirteen: Breaking the tls and dtls record protocols. In *IEEE Symposium on Security and Privacy* (2013), IEEE Computer Society, pp. 526–540.

[2] Brumley, D., and Boneh, D. Remote timing attacks are practical. *Computer Networks 48*, 5 (2005), 701–716.

[3] Chen, S., Wang, R., Wang, X., and Zhang, K. Side-channel leaks in web applications: A reality today, a challenge tomorrow. In *IEEE Symposium on Security and Privacy* (2010), IEEE Computer Society, pp. 191–206.

[4] Kocher, P. C. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *CRYPTO* (1996), N. Koblitz, Ed., vol. 1109 of *Lecture Notes in Computer Science*, Springer, pp. 104–113.

[5] Liu, W. M., Wang, L., Ren, K., Cheng, P., and Debbabi, M. k-indistinguishable traffic padding in web applications. In *Privacy Enhancing Technologies* (2012), S. Fischer-Hübner and M. Wright, Eds., vol. 7384 of *Lecture Notes in Computer Science*, Springer, pp. 79–99.

[6] Smith, B. Rizzo/Duong chosen plaintext attack (BEAST) on SSL/TLS 1.0 (facilitated by websockets -76). CVE-2011-3389,`https://bugzilla.mozilla.org/show_bug.cgi?id=665814`, 2011.

[7] T. Dierks, E. R. The Transport Layer Security(TLS) Protocol. `http://www.rfc-editor.org/rfc/rfc5246.txt`, 2008.

[8] Thai Duong, J. R. The CRIME Attack. `https://docs.google.com/presentation/d/11eBmGiHbYcHR9gL5nDyZChu_-lCa2GizeuOfaLU2HOU/`, 2012.

[9] Vaudenay, S. Security flaws induced by cbc padding - applications to ssl, ipsec, wtls ... In *EUROCRYPT* (2002), L. R. Knudsen, Ed., vol. 2332 of *Lecture Notes in Computer Science*, Springer, pp. 534–546.

[10] Yoel Gluck, Neal Harris, A. P. BREACH: REVIVING THE CRIME ATTACK. Blackhat USA ,`http://breachattack.com/resources/BREACH%20-%20SSL,%20gone%20in%2030%20seconds.pdf`, 2013.

[11] Zhang, Y., Juels, A., Reiter, M. K., and Ristenpart, T. Cross-vm side channels and their use to extract private keys. In *ACM Conference on Computer and Communications Security* (2012), T. Yu, G. Danezis, and V. D. Gligor, Eds., ACM, pp. 305–316.

Chapter 6

Appendix

6.1 Chen et. al. Attack

This attack demonstrates that plain text size is preserved after encryption and using statistical approach user query or activity on an web-application can be retrieved by attacker by profiling the web-application on all possible inputs. Depending on ciphertexts generated during user activity attacker could correspond to original plain texts that could have generated the monitored cipher texts. Authors modeled web-application as states and transition between states. More transition and corresponding packet generated reduced ambiguity. It was possible because of disparate sizes of possible inputs for application. They also attributed this weakness as a result of non-security aware design patterns followed to develop the application.

Detailed Attack

In this paper they modeled web applications as quintuple $(S, \Sigma, \delta, f, V)$.

S=Set of program states i.e application data on browser(DOM,Cookies),web server

Σ=Set of input i.e User inputs or Database

δ=Transition function from one program state to another

$(\delta : S \times \Sigma \rightarrow S)$

f=Observable web flows when transition takes place$(\delta : S \times \Sigma \rightarrow V)$,where V is a set of web flows. A web flow v is a sequence of directional packet sizes.

Objective of Adversary is to infer sensitive inputs that changes state of application. Consider an web application in which input set can be divided into different sets each of which can lead an application into different stage. This transition is accompanied by revelation of observable attributes.Consider at time t application is at state S_t. For any web application input set possible can be decomposed into k semantically disjoint sets each of which brings application into distinct state $S_{t+1} \subset S$. Each of this transition also reveals web flows v_t. Using this set

of web flows for each transition i.e $v_t, v_{t+1}, v_{t+2}, ..., v_{t+n-1}$ corresponding to each application state S_t attacker knows which input can possibly happen this transition and which might be the next program state. This method of reducing input set by observing attributes is *ambiguity reduction*. More flows attacker can see, more ambiguity he can reduce.Note here ambiguity is in inputs that might have caused the state transition. *reduction factor* is defined in this paper as

$$k/(\alpha\beta)$$

where,

k=Set of inputs possible for the application

α=Set of inputs identified by attacker by observing web flows and previous state.

β=Further reduction as program state continues to change depending on user inputs,based on web flows revealed in next state transitions.

Threats on an web application depends upon the size of disjoint input sets of an application i.e k. Today's web applications can be measured to be threat prone depending on above discussed parameters:

Ambiguity Reduction Factor indicates how much can the attacker learn from the observed web flows about the application. More he can learn less ambiguous it will be.

Input Size of Application indicates if it is efficient for attacker to profile the application for all possible input set and record it's web flows.

Since web applications are stateful. Thus each web flow vector generated from a transition increases reduces ambiguity factor since all transitions are dependent on previous transition.

Significant Traffic Distinctions for each web flow generated from a single transition attacker needs to take into account the objects that are exchanged between browser-server. For most applications distribution of such objects e.g. images,HTML code,JavaScript etc. are non-uniform thus leading to distinctions. This disparate size of objects are preserved due to the fact that underlying ciphers used are not built to mask the sizes without much overhead at HTTPS layer. Parameter to decide if web-application is vulnerable is **density**. Density determines how diverse the sizes of packets in web-application.

Mitigation

To mitigate this attack first thought that comes to mind is uniform distribution of objects. But at a protocol level its hard to do. Universal mitigation policies is a challenge. Application specific approaches are likely to mitigate such attack. In this paper they analyzed some possible universal mitigation policies on selected web applications.

The most obvious strategy is padding. Two types are considered here:

Random Padding. Appends every packet with a random length in $[0, \Delta)$.

*Rounding.*Round up size of every packet to a nearest multiple of Δ.

It is concluded that even after applying such mitigation it's not guaranteed that application will be information leakage free.Moreover they incur huge overhead.Application developers should understand application vulnerabilities first and then decide policies accordingly,since an application agnostic approach is unlikely to be feasible.

6.2 Other Mitigations

6.2.1 Liu et. al. Mitigation

This suggestion attempts to mitigate the above attack by providing a generic solution such that attacker cannot differentiate the inputs. In this approach authors suggested to group the inputs based on a specific privacy parameter k,such that all inputs in a group are padded to same size and padding cost is minimum. After such grouping response corresponding to the inputs belonging to same group will appear to have same size to attacker,thus attacker has problem in distinguishing the inputs. To make all packets corresponding to inputs same size padding is done with a padding scheme of choice.

Heuristic algorithms were designed which strikes a balance between privacy requirement and padding cost. This work was on application agnostic padding which is a more convenient approach if efficient,than application specific. Authors applied the concept of privacy preserving data publishing to mitigate this information leakage. They termed this padding scheme as *privacy preserving traffic padding*. This paper analyses side channel mitigation strategies in search engines.

The Model. In this paper web application is modeled in terms of observer's and interaction between users and servers. Each web interaction generates a flurry of packets to and from server. For some particular packets size varies in each interaction for e.g. due to different suggestion list sent by server. These size can be used to identify the input string.

Action. User activity upon application which generates observable traffic such as mouse click or keystroke.

Action Sequence. Related set of actions e.g. consecutive keystrokes.

Action Set. Set of all i'th action of all action sequences. For ex. If input='a' and '00' are two different action sequence. Here action is 'a' and '00'.Action set as per definition will be $A_1 = \{a, 0\}$ and $A_2 = \{0\}$.

Flow vector is set of all flows corresponding to an action.A flow vector sequence is a sequence of flow vector corresponding to each action in action sequence, each $V_{seq}[i]$ corresponding to action in $A_{seq}[i]$. For example,we have three flow vectors $v_1 = 508, v_2 = 497, v_3 = 578$ corresponding to action 'a 'and 'b(first keystroke),b(second keystroke)'.So,we have two vector se-

quences $v_1 and v_2 v_3$. We have two vector set $V_1 = \{508, 497\}$ and $V_2 = \{578\}$.Vector Action Set is defined as : Given an action set A_i and corresponding V_i, VA or vector action set is $\{(v, a) \mid v \in V_i \wedge a \in Ai\}$. For above ex. $VA_1 = \{(508, a), (497, b)\}$ and $VA_2 = \{(578, b)\}$.All algorithms works on vector action set. Web applications are modeled from simpler architecture to difficult and more real world depending on size of vector flow set and action set.Following algorithms were suggested based on the no. of web flows and no. of corresponding action sequence.

- SVSD can be used to model vector action set for traffic where there is only one action ,each action has only one flow in corresponding flow vector.
- SVMD can be used to model vector action set for traffic where there is only one action , each action has only multiple flow in corresponding flow vector.However for all actions in an action set has equal no of flows in a flow vector.Thus dummy packets needs to be inserted in case some has unequal no. of flows.
- In MVMD action in an action-set has multiple flows where multiple action are there in action sequence.

*k-anonymity.*This paper introduced k-anonymity, defined as the privacy requirement of the traffic or a given vector-action set.Basically,vector-action set is divided into k-sized groups based on minimum padding cost and all flow sizes in that group are same and indistinguishable from each other.Thus providing k-anonymity.Hence more k means more privacy given a cost.Padding in groups are done such that privacy cost i.e sum of distance between flows of each flow to maximum value in the group should be minimum satisfying the privacy requirement.For MVMD,Algorithms used the actions in same sequence to pad corresponding packets in same size since it puts those actions in same privacy group of size k.This is defined as prefix-based padding.