Norwegian University of Science and Technology

PADDING ORACLES

TTM4205 – Lecture 12

Tjerand Silde

17.10.2023

Contents

Contents

Reference Material

These slides are based on:

- The referred papers in the slides
- JPA: parts of chapter 4, 6 and 7
- DW: parts of chapter 2 to 4

Padding Oracles

By this we mean, on a high level, an API that allows an adversary to check if some input is correctly formed.

We limit ourselves to input with a particular padding.

A limited version of the protocol APIs from last week.

Padding Oracles

We will look at symmetric and asymmetric padding schemes:

- more in depth on CBC mode (today)
- extension attacks against hashing (today)
- padding attacks against RSA scheme (next)

Several of which are relevant to the weekly problems.

We will also look at some mitigations to these issues.

Contents

AES-CBC

Recall the CBC mode without authentication:

AES-CBC

Recall the CBC mode without authentication:

- Cipher mode for symmetric ciphers (e.g. AES)

AES-CBC

Recall the CBC mode without authentication:

- Cipher mode for symmetric ciphers (e.g. AES)
- Proposed in 1976, proven in 1997, broken 2002

AES-CBC

Recall the CBC mode without authentication:

- Cipher mode for symmetric ciphers (e.g. AES)
- Proposed in 1976, proven in 1997, broken 2002
- CPA secure (theory), not CCA (practice), patched

AES-CBC

Recall the CBC mode without authentication:

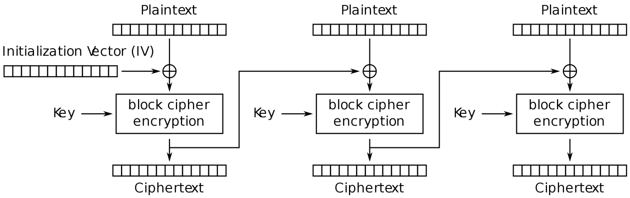
- Cipher mode for symmetric ciphers (e.g. AES)
- Proposed in 1976, proven in 1997, broken 2002
- CPA secure (theory), not CCA (practice), patched
- A variety of padding oracle attacks in practice

AES-CBC

Recall the CBC mode without authentication:

- Cipher mode for symmetric ciphers (e.g. AES)
- Proposed in 1976, proven in 1997, broken 2002
- CPA secure (theory), not CCA (practice), patched
- A variety of padding oracle attacks in practice
- Revoked from some applications (e.g. TLS) in 2018

AES-CBC



Cipher Block Chaining (CBC) mode encryption

CBC Attack

CBC Attack

- Each block must be of exactly 128 bits

CBC Attack

- Each block must be of exactly 128 bits
- Shorter message leads to padding at the end

CBC Attack

- Each block must be of exactly 128 bits
- Shorter message leads to padding at the end
- Add one byte ends with 01, two with 02, etc. ...

CBC Attack

- Each block must be of exactly 128 bits
- Shorter message leads to padding at the end
- Add one byte ends with 01, two with 02, etc. ...
- An API outputs errors when wrong padding

CBC Attack

CBC Attack

- Let C_2 be an encryption of X that you want to decrypt

CBC Attack

- Let C_2 be an encryption of X that you want to decrypt
- Choose random C_1 and ask for $C_1|C_2$ to be decrypted

CBC Attack

- Let C_2 be an encryption of X that you want to decrypt
- Choose random C_1 and ask for $C_1|C_2$ to be decrypted
- Successful decryption if $C_1 \oplus X$ has valid padding

CBC Attack

- Let C_2 be an encryption of X that you want to decrypt
- Choose random C_1 and ask for $C_1|C_2$ to be decrypted
- Successful decryption if $C_1 \oplus X$ has valid padding
- Vary last byte of C_1 until correct to find last byte of X

CBC Attack

- Let C_2 be an encryption of X that you want to decrypt
- Choose random C_1 and ask for $C_1|C_2$ to be decrypted
- Successful decryption if $C_1 \oplus X$ has valid padding
- Vary last byte of C_1 until correct to find last byte of X
- Find next byte by $C_1[15] = X[15] \oplus 02$ and vary $C_1[14]$

CBC Attack

- Let C_2 be an encryption of X that you want to decrypt
- Choose random C_1 and ask for $C_1|C_2$ to be decrypted
- Successful decryption if $C_1 \oplus X$ has valid padding
- Vary last byte of C_1 until correct to find last byte of X
- Find next byte by $C_1[15] = X[15] \oplus 02$ and vary $C_1[14]$
- Continue until you have all bytes of X, max $128 \cdot 16$ trials

CBC Attack

Cryptopals: Exploiting CBC Padding Oracles

This is a write-up of the classic padding oracle attack on CBC-mode block ciphers. If you've done the Cryptopals cryptography challenges, you'll remember it as challenge 17. This is a famous and elegant attack. With it, we will see how even a small data leak (in this case, the presence of a "padding oracle" – defined below) can lead to full plaintext recovery.

Like the Cryptopals challenges, this post is written to be accessible to anyone with an interest in cryptography – no graduate degree required. All you need is patience, focus, and some basic familiarity with the concepts in the following section.

Figure: `https://research.nccgroup.com/2021/02/17/cryptopals-exploiting-cbc-padding-oracles`

CBC Attack

Figure: https://blog.cryptographyengineering.com/2011/10/23/attack-of-week-xml-encryption

Contents

Authenticated CBC Mode

If we check that the CBC encryption was correctly computed, then we do not need to worry about the padding oracle.

Question 1: How to securely use CBC mode with MAC?

Authenticated CBC Mode

If we check that the CBC encryption was correctly computed, then we do not need to worry about the padding oracle.

Question 2: What are possible mitigations for CBC?

Possible Mitigations

- Randomized padding scheme
- Fixed size padding
- Additional randomized delay
- No specific error message

Question: What might go wrong in these cases?

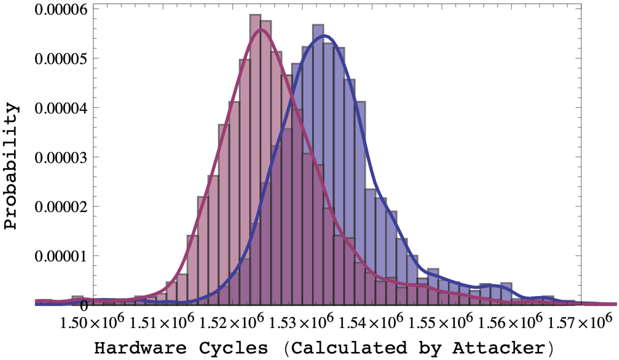


Figure 2. Distribution of timing values (outliers removed) for distinguishing attack on OpenSSL TLS, showing faster processing time in the case of M_0 (in red) compared to M_1 (in blue).

Figure: https://www.ieee-security.org/TC/SP2013/papers/4977a526.pdf

General Solutions

- Always use authenticated encryption
- Avoid CBC mode if possible (use GCM)
- Constant time padding check
- No specific error messages

2013 IEEE Symposium on Security and Privacy

Lucky Thirteen: Breaking the TLS and DTLS Record Protocols

Nadhem J. AlFardan and Kenneth G. Paterson
Information Security Group,
Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK
Email: {nadhem.alfardan.2009, kenny.paterson}@rhul.ac.uk

Figure: https://www.ieee-security.org/TC/SP2013/papers/4977a526.pdf

Contents

Hashing as MAC

Assume that we are in the following setting:

- Let sk be a fixed size secret
- Let m be a known message
- Let H be a the SHA2 hash function
- Let MAC be $h = H(\mathsf{sk}||m)$

Question 1: Do you remember how SHA2 works?

Hashing as MAC

Assume that we are in the following setting:

- Let sk be a fixed size secret
- Let m be a known message
- Let H be a the SHA2 hash function
- Let MAC be $h = H(\mathsf{sk}||m)$

Question 2: How can we forge $h' = H(\mathsf{sk}||m')$?

Hashing as MAC

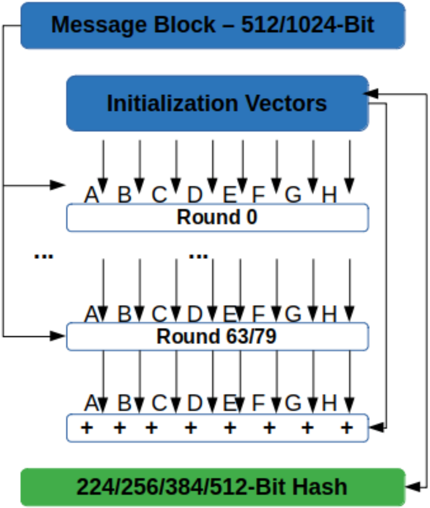
The issue is that SHA2 apply a compression function on blocks of the message using length padding in the end.

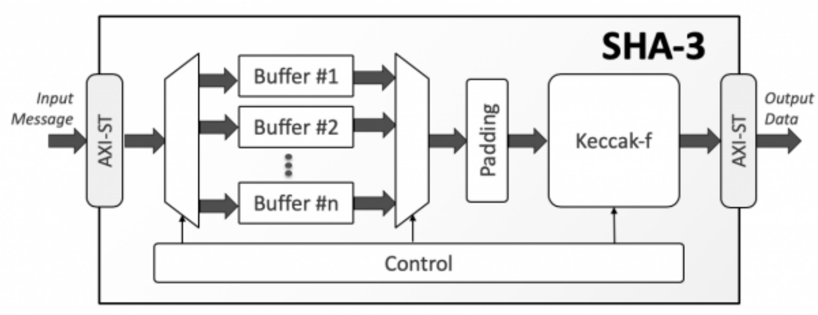
If you know the length of the secret and the message, then you also know the padding, and you can append a message at the end to get a valid hash without knowing the secret.

Hashing as MAC

This attack applies to SHA2, but not to SHA3. SHA3 has a different structure. Does not apply to HMAC using SHA2.

SHA-2 Process Overview




SHA-3
Input
Message
AXI-ST
Buffer #1
Buffer #2
Buffer #n
Padding
Keccak-f
AXI-ST
Output
Data
Control

Length extension attack

Deep RnD · Follow
4 min read · Aug 16, 2019

12 1

What is length extension?

When a Merkle-Damgård based hash is misused as a message authentication code with construction H(secret ‖ message), and message and the length of secret is known, a length extension attack allows anyone to include extra information at the end of the message and produce a valid hash without knowing the secret. Quick sidebar, before you freak out:

> Since HMAC does not use this construction, HMAC hashes are not prone to length extension attacks.

Figure: `https://deeprnd.medium.com/length-extension-attack-bff5b1ad2f70`

Contents

Order of Enc and Auth

CPA secure symmetric crypto with input message m:

- SSL/TLS: $a = \mathsf{Auth}_{\mathsf{sk_A}}(\mathsf{m}), \mathsf{c} = \mathsf{Enc}_{\mathsf{sk_E}}(\mathsf{a}, \mathsf{m}), \text{send c}$
- IPSec: $c = \mathsf{Enc}_{\mathsf{sk_E}}(\mathsf{m}), \mathsf{a} = \mathsf{Auth}_{\mathsf{sk_A}}(\mathsf{c}), \text{send } (\mathsf{a}, \mathsf{c})$
- SSH: $c = \mathsf{Enc}_{\mathsf{sk_E}}(\mathsf{m}), \mathsf{a} = \mathsf{Auth}_{\mathsf{sk_A}}(\mathsf{m}), \text{send } (\mathsf{a}, \mathsf{c})$

Order of Enc and Auth

We refer to these methods as:

- SSL/TLS: authenticate-then-encrypt (AtE)
- IPSec: encrypt-then-authenticate (EtA)
- SSH: encrypt-and-authenticate (E&A)

Order of Enc and Auth

We refer to these methods as:

- SSL/TLS: authenticate-then-encrypt (AtE) (*can* be secure)
- IPSec: encrypt-then-authenticate (EtA) (proven secure)
- SSH: encrypt-and-authenticate (E&A) (shown broken)

Interestingly, AtE is proven secure when using CBC mode.

The Order of Encryption and Authentication for Protecting Communications (Or: How Secure is SSL?)*

Hugo Krawczyk**

Figure: https://iacr.org/archive/crypto2001/21390309.pdf

Questions?