NTNU | Norwegian University of Science and Technology

PADDING ORACLES

TTM4205 – Lecture 13

Tjerand Silde

19.10.2023

Contents

Contents

Reference Material

These slides are based on:

- The referred papers in the slides
- JPA: parts of chapter 10
- DW: parts of chapter 6

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 (last time)
- extension attacks against hashing (last time)
- padding attacks against RSA scheme (today)

Several of which are relevant to the weekly problems.

We will also look at some mitigations to these issues.

Contents

Textbook RSA

The plain RSA encryption scheme works as follows:

KGen:

- Samples primes p and q of appropriate size and entropy
- Use fixed e and compute $d \equiv e^{-1} \bmod \operatorname{lcm}(p-1, q-1)$
- Output the key pair $\mathsf{pk} = (e, n = p \cdot q)$ and $\mathsf{sk} = (d, p, q)$

Textbook RSA

The plain RSA encryption scheme works as follows:

Enc:

- Takes as input a message m and public key $\mathsf{pk} = (e, n)$
- Computes the ciphertext $c \equiv m^e \bmod n$ and outputs c

Textbook RSA

The plain RSA encryption scheme works as follows:

Dec:

- Takes as input a ciphertext c and secret key $\mathsf{sk} = (d, p, q)$
- Computes the message $m \equiv c^d \bmod p \cdot q$ and outputs m

Textbook RSA

Question: Why is not the textbook RSA scheme secure?

Textbook RSA

The following things make the RSA scheme insecure:

- It is not randomized and hence not even CPA secure
- Given a ciphertext you can search for the message
- High-entropy messages still gives the same ciphertext
- The Jacobi symbol of m and c will be the same

Solution: structured, but randomized padding

Contents

RSA-PKCS#1v1.5

Let n be of k bytes. Given a message m of $\ell \leq k - 11$ bytes, the padded messages $\bar{m}$ of length k bytes is constructed as follows: $00\ 02$ {at least 8 non-zero random bytes} 00 $\{m\}$.

Quite simple, not proven secure, not secure in practice...

Matthew Green in attacks, tokens/HSMs · June 21, 2012 · 2,313 Words

A bad couple of years for the cryptographic token industry

SafeNet eToken PRO Anywhere

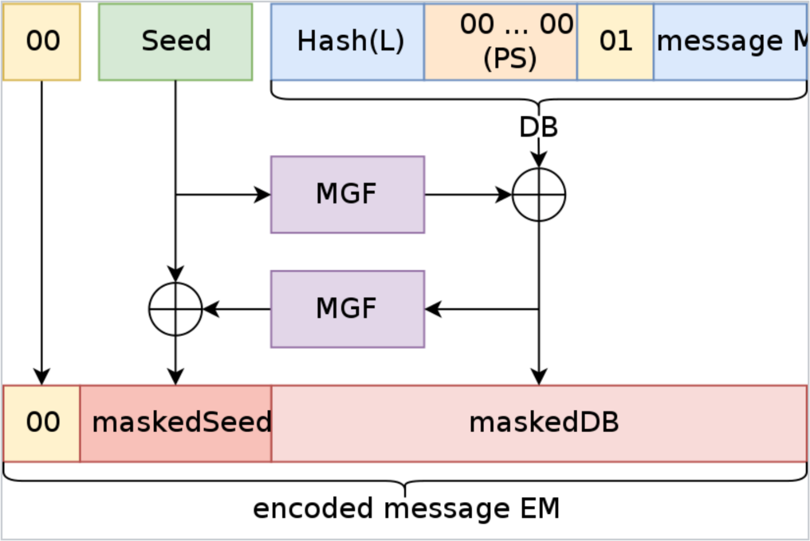
Figure: https://blog.cryptographyengineering.com/2012/06/21/bad-couple-of-years-for-cryptographic

OAEP

More complex, proven secure, what you should use:

- Let n be of k bytes and message m be of ℓ bytes.
- Let MGF and Hash be hash functions with output h bytes.
- Let L be a label (which can be set to the all zero string)
- Let seed be an ephemeral random string of h bytes.
- Let PS be a all zero string of length $k - \ell - 2h - 2$ bytes.

OAEP


00
Seed
Hash(L)
00 ... 00
(PS)
01
message M
DB
MGF
MGF
00
maskedSeed
maskedDB
encoded message EM

Optimal Asymmetric Encryption — How to Encrypt with RSA

Mihir Bellare* Phillip Rogaway†

November 19, 1995

Figure: https://cseweb.ucsd.edu/~mihir/papers/oaep.pdf

Contents


New cryptographers
Experienced cryptographers

The Bleichenbacher Attack

However, many implementations (still) use RSA-PKCS#1v1.5 or similar padding schemes (note that this is version 1.5).

Recall: $00\ 02\ \{$at least 8 non-zero random bytes$\}\ 00\ \{m\}$.

Question: Assuming no integrity check of RSA ciphertexts, how could you attack this scheme?

The Bleichenbacher Attack

The Bleichenbacher Attack

- Recall that RSA is homomorphic: $\bar{m}^e \cdot r^e \equiv (\bar{m} \cdot r)^e \bmod n$.

The Bleichenbacher Attack

- Recall that RSA is homomorphic: $\bar{m}^e \cdot r^e \equiv (\bar{m} \cdot r)^e \bmod n$.
- We can multiply a ciphertext c by a chosen $r^e \bmod n$.

The Bleichenbacher Attack

- Recall that RSA is homomorphic: $\bar{m}^e \cdot r^e \equiv (\bar{m} \cdot r)^e \bmod n$.
- We can multiply a ciphertext c by a chosen $r^e \bmod n$.
- We learn if padding is valid (error message or timings).

The Bleichenbacher Attack

- Recall that RSA is homomorphic: $\bar{m}^e \cdot r^e \equiv (\bar{m} \cdot r)^e \bmod n$.
- We can multiply a ciphertext c by a chosen $r^e \bmod n$.
- We learn if padding is valid (error message or timings).
- We know the value r since we could choose it ourselves.

The Bleichenbacher Attack

- Recall that RSA is homomorphic: $\bar{m}^e \cdot r^e \equiv (\bar{m} \cdot r)^e \bmod n$.
- We can multiply a ciphertext c by a chosen $r^e \bmod n$.
- We learn if padding is valid (error message or timings).
- We know the value r since we could choose it ourselves.
- Then we learn that unknown $\bar{m}$ times r is valid encoding.

The Bleichenbacher Attack

- Recall that RSA is homomorphic: $\bar{m}^e \cdot r^e \equiv (\bar{m} \cdot r)^e \bmod n$.
- We can multiply a ciphertext c by a chosen $r^e \bmod n$.
- We learn if padding is valid (error message or timings).
- We know the value r since we could choose it ourselves.
- Then we learn that unknown $\bar{m}$ times r is valid encoding.
- We know that if valid then $2 \cdot 2^{8(k-2)} \leq \bar{m} \cdot r < 3 \cdot 2^{8(k-2)}$.

The Bleichenbacher Attack

- Recall that RSA is homomorphic: $\bar{m}^e \cdot r^e \equiv (\bar{m} \cdot r)^e \bmod n$.
- We can multiply a ciphertext c by a chosen $r^e \bmod n$.
- We learn if padding is valid (error message or timings).
- We know the value r since we could choose it ourselves.
- Then we learn that unknown $\bar{m}$ times r is valid encoding.
- We know that if valid then $2 \cdot 2^{8(k-2)} \leq \bar{m} \cdot r < 3 \cdot 2^{8(k-2)}$.
- Repeat for fresh values r until we have a unique $\bar{m}$ left.

Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1

Daniel Bleichenbacher

Bell Laboratories
700 Mountain Ave.
Murray Hill, NJ 07974
E-mail: bleichen@research.bell-labs.com

Figure: https://spar.isi.jhu.edu/~mgreen/bleichenbacher.pdf

Contents

Improving the Attack

Improving the Attack

1. Be clever when choosing r using co-prime samples.

Improving the Attack

1. Be clever when choosing r using co-prime samples.
2. Trim the randomness to a specific interval $[a, b]$

Improving the Attack

1. Be clever when choosing r using co-prime samples.
2. Trim the randomness to a specific interval $[a, b]$
3. Parallelization and threading and pre-computation

Improving the Attack

1. Be clever when choosing r using co-prime samples.
2. Trim the randomness to a specific interval $[a, b]$
3. Parallelization and threading and pre-computation
4. Adapt based on how strict padding checks are

Padding Checks

The efficiency depends on how strict the padding check is:

Padding Checks

The efficiency depends on how strict the padding check is:

1. FFF: padding is ‘ok’ only if correctly padded and plaintext is of a specific length (e.g., it’s a 128-bit AES key).

Padding Checks

The efficiency depends on how strict the padding check is:

1. FFF: padding is ‘ok’ only if correctly padded and plaintext is of a specific length (e.g., it’s a 128-bit AES key).

2. FFT: padding is ‘ok’ only if correctly padded, but plaintext can be any length.

Padding Checks

The efficiency depends on how strict the padding check is:

1. FFF: padding is 'ok' only if correctly padded and plaintext is of a specific length (e.g., it's a 128-bit AES key).

2. FFT: padding is 'ok' only if correctly padded, but plaintext can be any length.

3. FTT: same as above, but also allows 0s in the "non-zero random bytes".

Padding Checks

The efficiency depends on how strict the padding check is:

1. FFF: padding is ‘ok’ only if correctly padded and plaintext is of a specific length (e.g., it’s a 128-bit AES key).

2. FFT: padding is ‘ok’ only if correctly padded, but plaintext can be any length.

3. FTT: same as above, but also allows 0s in the “non-zero random bytes”.

4. TFT: same as above, but ‘ok’ even if there are no zeros after the first byte.

Padding Checks

The efficiency depends on how strict the padding check is:

1. FFF: padding is 'ok' only if correctly padded and plaintext is of a specific length (e.g., it's a 128-bit AES key).

2. FFT: padding is 'ok' only if correctly padded, but plaintext can be any length.

3. FTT: same as above, but also allows 0s in the "non-zero random bytes".

4. TFT: same as above, but 'ok' even if there are no zeros after the first byte.

5. TTT: padding is 'ok' as long as it starts with 0x 00 02.

Oracle	Original algorithm		Modified algorithm			
	Mean	Median	Mean	Median	Trimmers	Mean skipped
FFF	-	-	18 040 221	12 525 835	50 000	7 321
FFT	215 982	163 183	49 001	14 501	1 500	65 944
FTT	159 334	111 984	39 649	11 276	2 000	61 552
TFT	39 536	24 926	10 295	4 014	600	20 192
TTT	38 625	22 641	9 374	3 768	500	18 467

Table 1: Performance of the original and modified algorithms.

Figure: https://eprint.iacr.org/2012/417.pdf

Efficient Padding Oracle Attacks on Cryptographic Hardware[⋆]

Romain Bardou[1], Riccardo Focardi[2⋆⋆], Yusuke Kawamoto[3⋆⋆⋆],
Lorenzo Simionato[2†], Graham Steel[4⋆⋆⋆], and Joe-Kai Tsay[5⋆⋆⋆]

[1] INRIA SecSI, LSV, CNRS & ENS-Cachan, France
[2] DAIS, Università Ca' Foscari, Venezia, Italy
[3] School of Computer Science, University of Birmingham, UK
[4] INRIA Project ProSecCo, Paris, France
[5] Department of Telematics, NTNU, Norway

Figure: https://eprint.iacr.org/2012/417.pdf

Contents

Protection

Protection

- Use OAEP padding for encryption

Protection

- Use OAEP padding for encryption
- Encrypt-then-Authenticate

Protection

- Use OAEP padding for encryption
- Encrypt-then-Authenticate
- Do not use RSA for encryption

Questions?