

CBC padding oracle attacks  
CSE 468 Fall 2024  
jedimaestro@asu.edu

Some good motivational reading:

<https://citizenlab.ca/2023/08/vulnerabilities-in-sogou-keyboard-encryption/>

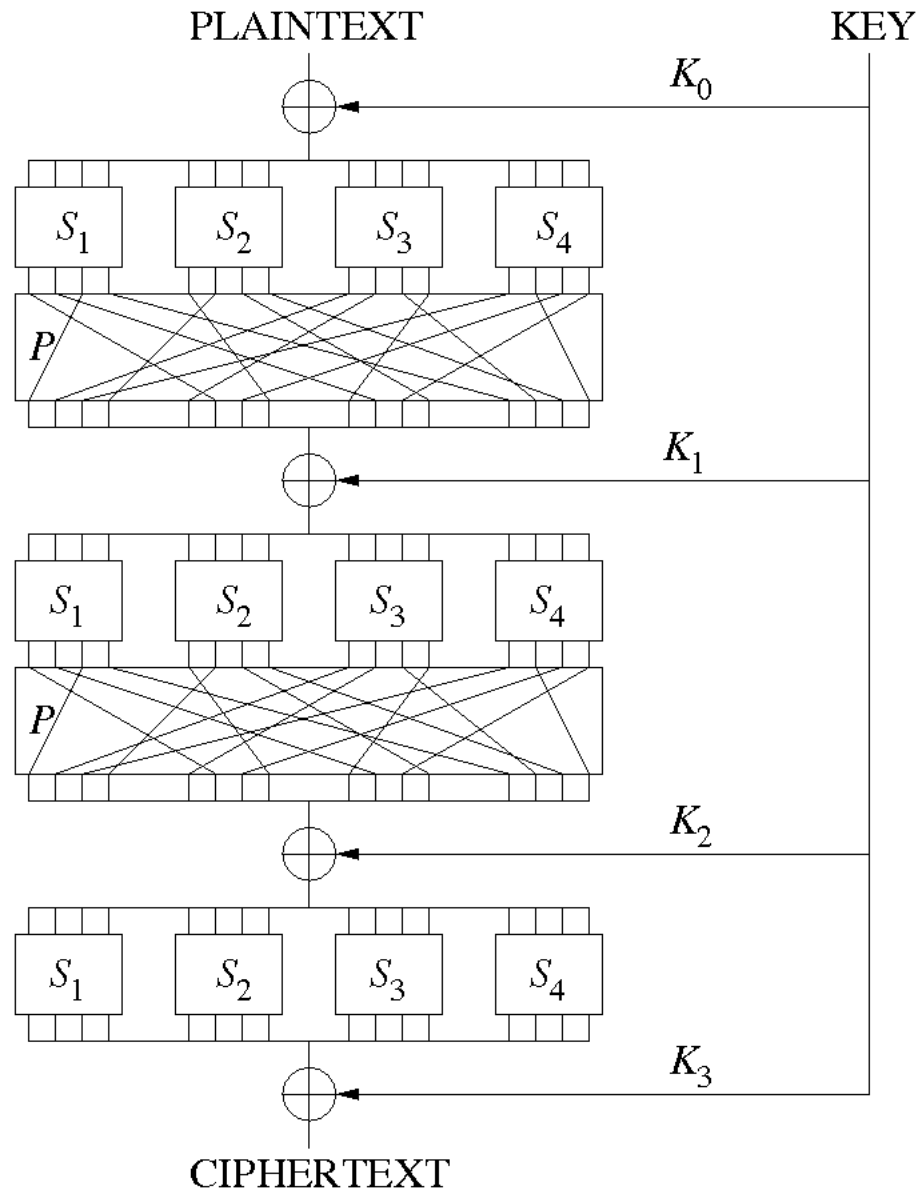
# Midterm review

- **Homework with info theory and modular exponentiation**
- In-path vs. on-path vs. off-path
- AES S-Box requirements
- Properties of XOR
- Secure hash functions
  - Three properties (slides 28-30)
  - Three basic attacks (slides 32, 36, 38)
- ECB vs. CBC
- Block ciphers vs. stream ciphers (Thursday)
- Basics of asymmetric crypto vs. symmetric crypto

# CBC padding oracle attack examples

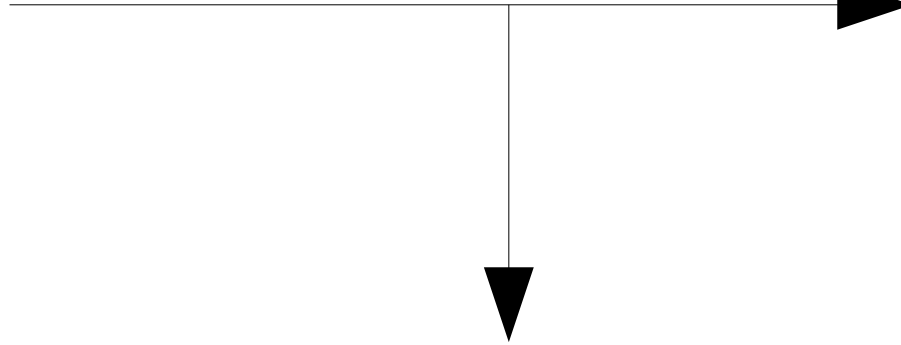
- Serge Vaudenay published the original attack in 2002
  - Applied to web frameworks like Ruby on Rails, ASP.NET, and JavaServer Faces
  - <https://www.iacr.org/cryptodb/archive/2002/EUROCRYPT/2850/2850.pdf>
- POODLE (published by Google in 2014) exploited SSLv3 that is still widely used by web servers and browsers
  - <https://security.googleblog.com/2014/10/this-poodle-bites-exploiting-ssl-30.html>

# Review: AES is a block cipher and is symmetric



Source: Wikipedia

# Alice and Bob have a shared secret key



Eve makes a copy of the ciphertext as it is transmitted from Alice to Bob.

# Alice and Bob have a shared secret key



Eve re-plays modified copies of the encrypted message and learns information about the plaintext from Bob's behavior (e.g., Bob throws an exception for padding error)

# PKCS#7 padding

- AES always encrypts in 128-bit blocks
  - 128 bits == 16 bytes
- If you fill up blocks, that's great
  - But, the last block might not be full
- Need an “unambiguous” way to pad the last block so the decrypting party knows the padding to throw out
  - *E.g.*, PKCS#7 (PKCS == Public Key Cryptography Standards)

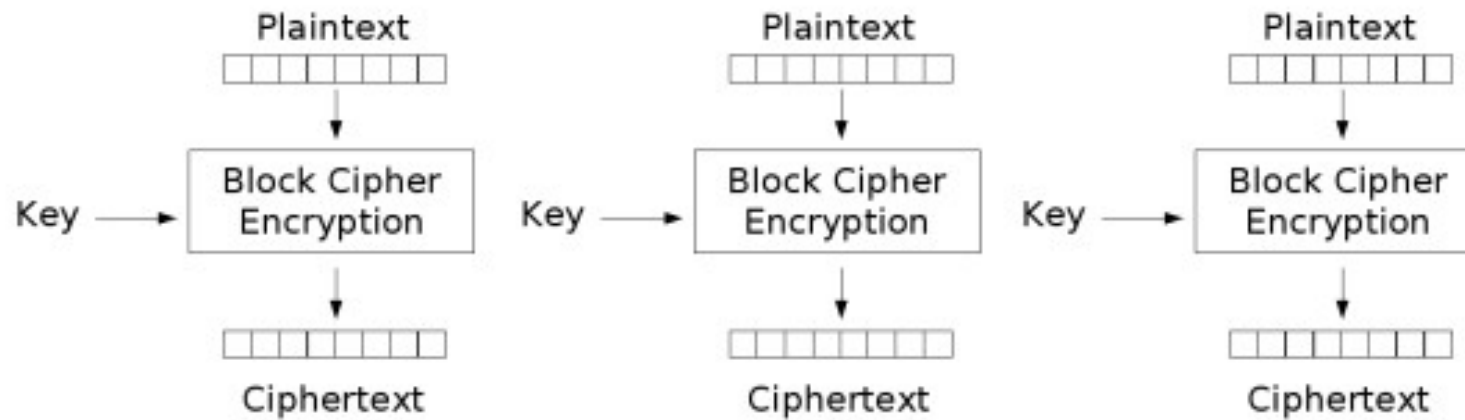




# When last block is decrypted

- Check last byte of the last block, that's the number of bytes of padding
  - Call it N
- There should be N N's on the end
  - If not, throw a padding error
  - If so, remove them, they're padding
    - Might remove the whole last block if  $N = 16$  (or 10 in hex)

# Electronic Codebook (ECB)



Electronic Codebook (ECB) mode encryption

Image stolen from Wikipedia

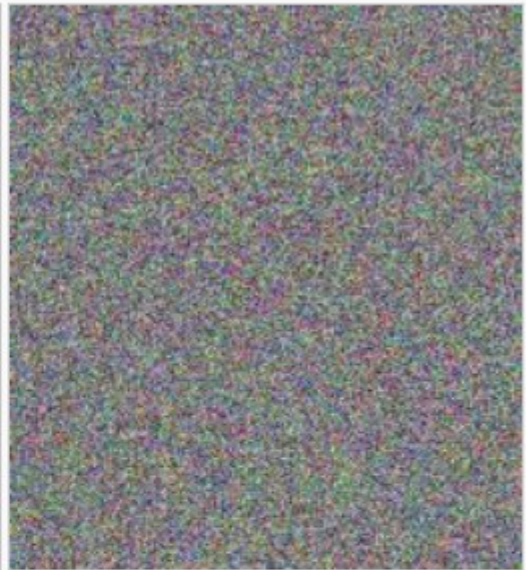
# ECB is generally bad



Original image



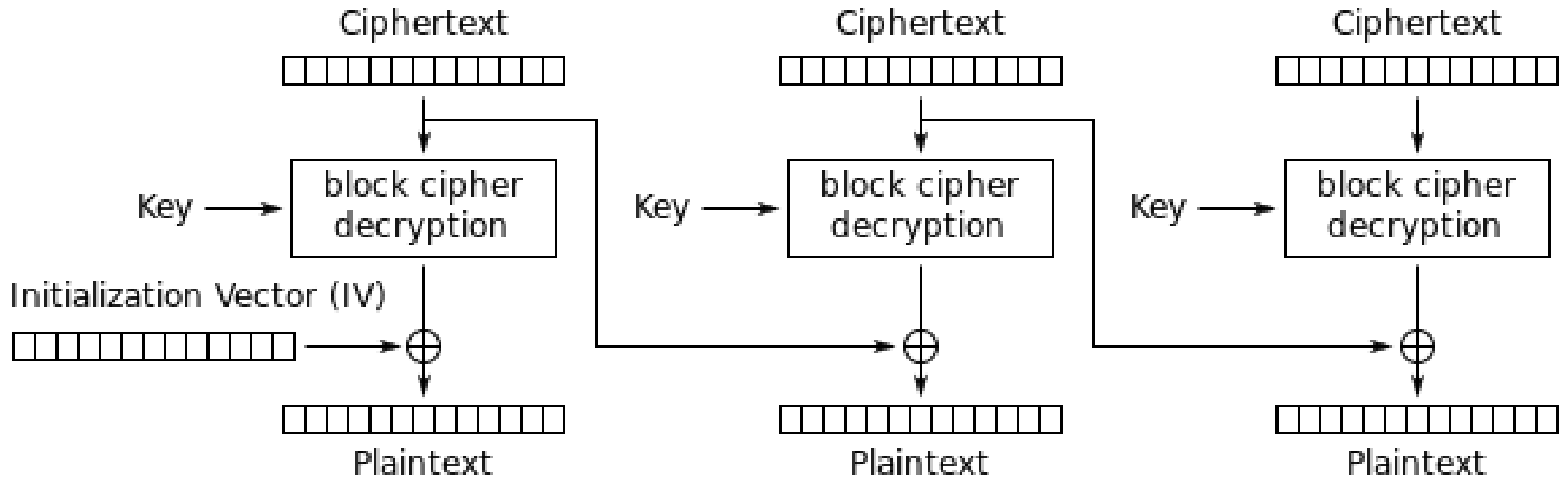
Encrypted using ECB mode



Modes other than ECB result in pseudo-randomness

The image on the right is how the image might appear encrypted with CBC, CTR or any of the other more secure modes—indistinguishable from random noise. Note that the random appearance of the image on the right does not ensure that the image has been securely encrypted; many kinds of insecure encryption have been developed which would produce output just as "random-looking".

Image stolen from Wikipedia



## Cipher Block Chaining (CBC) mode decryption

# Requirements for attack

- Ability to modify ciphertexts and replay them
  - Chosen ciphertext attack
- A padding oracle
  - *i.e.*, something that tells you whether the corresponding plaintext (for any ciphertext you send) has valid padding or not

# Example plaintext (we don't know the plaintext yet before the attack)

H	e	l	l	o	20	W	o	r	l	d	!	\n	03	03	03
---	---	---	---	---	----	---	---	---	---	---	---	----	----	----	----


# Example protocol for a client to send an encrypted message to a server

N	u	m	b	l	k	s	:	1	K	e	y	l	D	:	A3
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
98	CC	BE	01	FF	26	39	97	85	A1	02	1E	BC	A5	7E	98

# Example protocol for a client to send an encrypted message to a server

Number of blocks

Which key?



N	u	m	b	l	k	s	:	1	K	e	y	I	D	:	A3
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
98	CC	BE	01	FF	26	39	97	85	A1	02	1E	BC	A5	7E	98



# Example protocol for a client to send an encrypted message to a server

N	u	m	B	l	k	s	:	1	K	e	y	l	D	:	A3
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
98	CC	BE	01	FF	26	39	97	85	A1	02	1E	BC	A5	7E	98

IV is randomly chosen but visible on the wire and known to you, won't be 0 like in this illustration

# Example protocol for a client to send an encrypted message to a server

N	u	m	B	l	k	s	:	1	K	e	y	l	D	:	A3
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
98	CC	BE	01	FF	26	39	97	85	A1	02	1E	BC	A5	7E	98

Ciphertext is what you want to decrypt, you will recover the plaintext without needing to know the key!

# Server response is visible to you

- “Message decrypted successfully”  
---or---
- “Padding error during decryption”

# You can record a client message and replay it to the server

N	u	m	b	l	k	s	:	1	K	e	y	l	D	:	A3
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
98	CC	BE	01	FF	26	39	97	85	A1	02	1E	BC	A5	7E	98

Try every value of this byte from 00 to FF

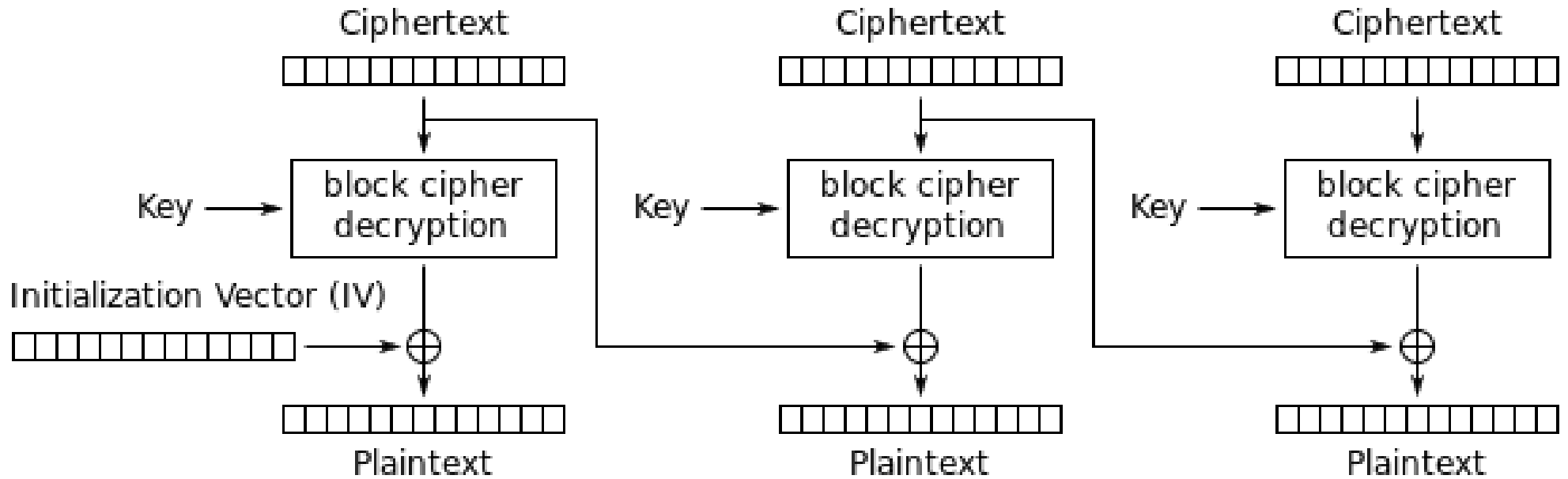


# You can record a client message and replay it to the server

N	u	m	b	l	k	s	:	1	K	e	y	l	D	:	A3
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
98	CC	BE	01	FF	26	39	97	85	A1	02	1E	BC	A5	7E	98

Try every value of this byte from 00 to FF, will flip bits here...

H	e	l	l	o	20	W	o	r	l	d	!	\n	03	03	03
---	---	---	---	---	----	---	---	---	---	---	---	----	----	----	----



## Cipher Block Chaining (CBC) mode decryption

# Suppose two values give valid padding

- 00 gives valid padding, this is just confirmation that the original plaintext has valid padding
  - 02 also gives valid padding
    - Can recover one byte of plaintext:  
 $Q \text{ XOR } 02 == 01$ , so...  $Q == 01 \text{ XOR } 02 == 03$
- Q is the byte of plaintext we're trying to guess

# WTF?

N	u	m	b	l	k	s	:	1	K	e	y	l	D	:	A3
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	02
98	CC	BE	01	FF	26	39	97	85	A1	02	1E	BC	A5	7E	98



H	e	l	l	o	20	W	o	r	l	d	!	\n	03	03	01
---	---	---	---	---	----	---	---	---	---	---	---	----	----	----	----

“Information only has meaning in that it is subject to interpretation”



$$01 \text{ XOR } 02 = 03$$

N	u	m	b	l	k	s	:	1	K	e	y	l	D	:	A3
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	01
98	CC	BE	01	FF	26	39	97	85	A1	02	1E	BC	A5	7E	98



H	e	l	l	o	20	W	o	r	l	d	!	\n	03	03	02
---	---	---	---	---	----	---	---	---	---	---	---	----	----	----	----



Now attack here

$$01 \text{ XOR } 02 = 03$$

Hold this at 01

N	u	m	b	l	k	s	:	1	K	e	y	l	D	:	A3
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	01
98	CC	BE	01	FF	26	39	97	85	A1	02	1E	BC	A5	7E	98



H	e	l	l	o	20	W	o	r	l	d	!	\n	03	03	02
---	---	---	---	---	----	---	---	---	---	---	---	----	----	----	----

Now attack here

# Discussion

- You still don't know the key, probably never will
- It doesn't matter how secure AES is or the size of the key
- CBC is probably the most commonly used mode for some application types
- What if a byte is already what it needs to be?
- What if there is more than one block?

# References

- <https://grymoire.wordpress.com/2014/12/05/cbc-padding-oracle-attacks-simplified-key-concepts-and-pitfalls/>