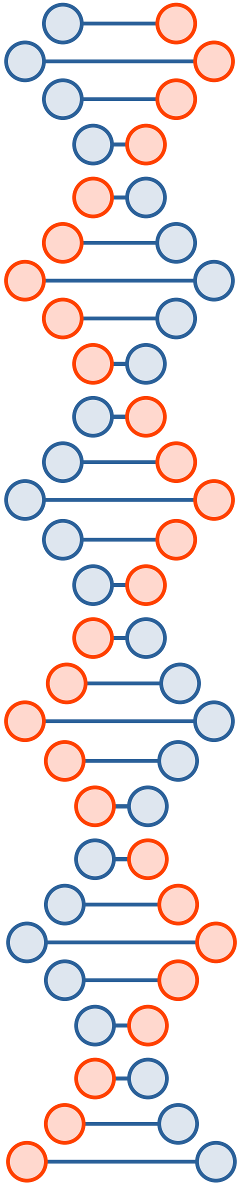
AES and cipherblock chaining modes

CSE 468 Fall 2025
jedimaestro@asu.edu

Why study AES?

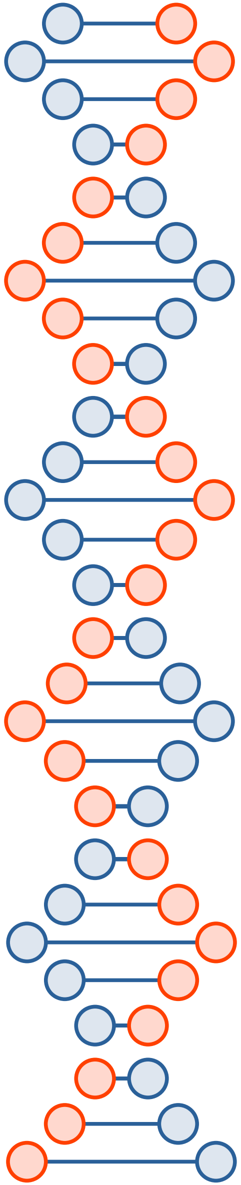
- It's a good demonstration of principles we've been talking about (*e.g.*, confusion and diffusion)
- It's the workhorse of the majority of network crypto, *e.g.*, many SSH and TLS connections
- It's easy to see that AES is just substitution, permutation, and XOR



Substitution

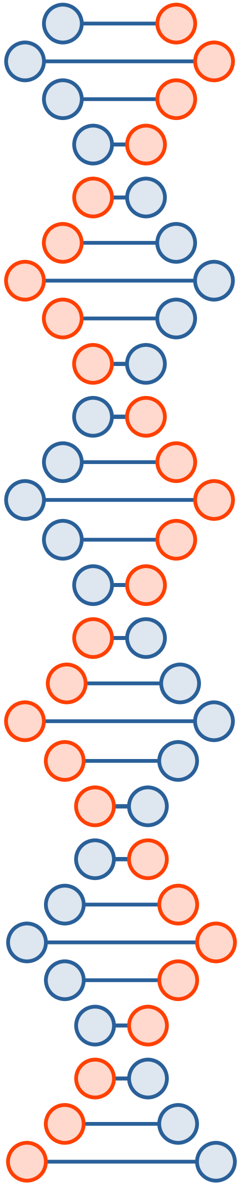
HELLO WORLD

TNWWX DXPWE



Permutation

ABCD	ABDC	ACBD	ACDB	ADBC	ADCB
BACD	BADC	BCAD	BCDA	BDAC	BDCA
CABD	CADB	CBAD	CBDA	CDAB	CDBA
DABC	DACB	DBAC	DBCA	DCAB	DCBA



Bitwise XOR

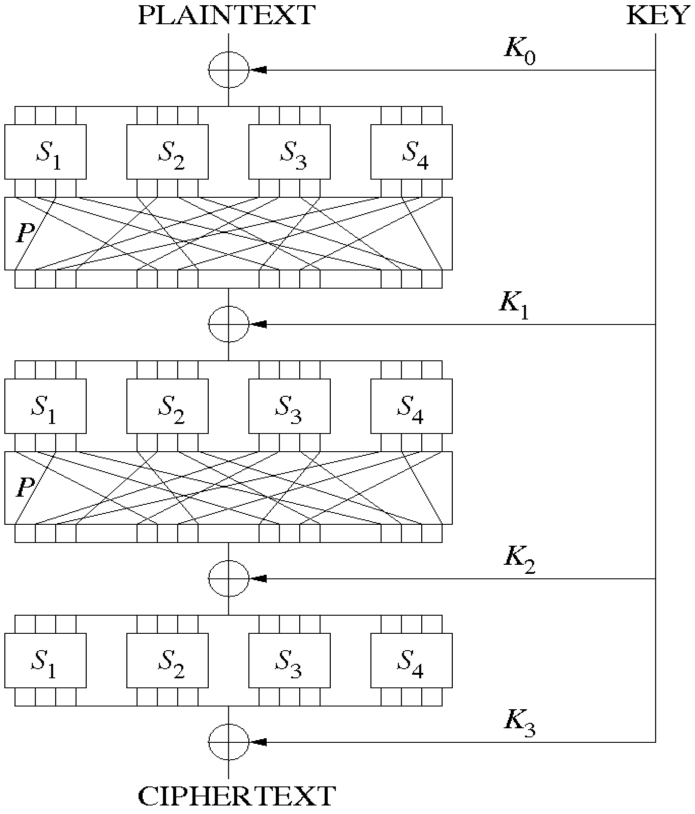
$$
\begin{array}{r}
00101010_b \\
\oplus 10000110_b \\
= 10101100_b
\end{array}
$$

Symmetric encryption over time...

- Handwritten notes, *etc.* for centuries
  - Typically the algorithm was secret
- 1883 … Kerckhoff's rules
  - Now we know the key should be the only secret
- 1975 … DES
  - Efficient in hardware, not in software
- 2001 … AES
  - Efficient in software, and lots of different kinds of hardware

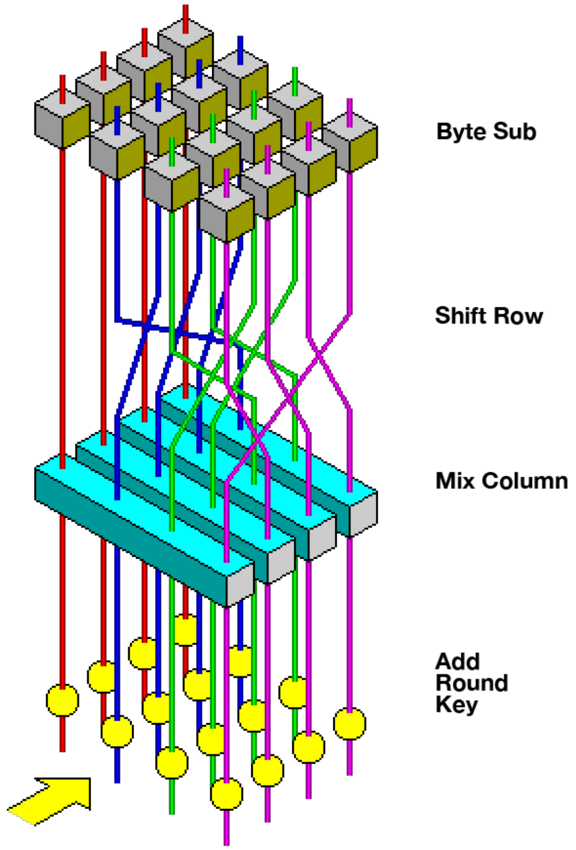
Substitution Permutation Network

e.g., AES 128-bit blocks, (128-, 192-, 256-)bit key, (10, 12, 14) rounds



AES

- Rijndael
  - Joan Daemen and Vincent Rijmen
- Very clever S-box design that comes from Kaisa Nyberg
  - Based on finite fields (*a.k.a.* Gallois fields)



⊗	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	0	2	4	6	8	A	C	E	3	1	7	5	B	9	F	D
3	0	3	6	5	C	F	A	9	B	8	D	E	7	4	1	2
4	0	4	8	C	3	7	B	F	6	2	E	A	5	1	D	9
5	0	5	A	F	7	2	D	8	E	B	4	1	9	C	3	6
6	0	6	C	A	B	D	7	1	5	3	9	F	E	8	2	4
7	0	7	E	9	F	8	1	6	D	A	3	4	2	5	C	B
8	0	8	3	B	6	E	5	D	C	4	F	7	A	2	9	1
9	0	9	1	8	2	B	3	A	4	D	5	C	6	F	7	E
A	0	A	7	D	E	4	9	3	F	5	8	2	1	B	6	C
B	0	B	5	E	A	1	F	4	7	C	2	9	D	6	8	3
C	0	C	B	7	5	9	E	2	A	6	1	D	F	3	4	8
D	0	D	9	4	1	C	8	5	2	F	B	6	3	E	A	7
E	0	E	F	1	D	3	2	C	9	7	6	8	4	A	B	5
F	0	F	D	2	9	6	4	8	1	E	C	3	8	7	5	A

An alternative to AES: Tiny Encryption Algorithm (TEA), Feistel structure with 32 rounds

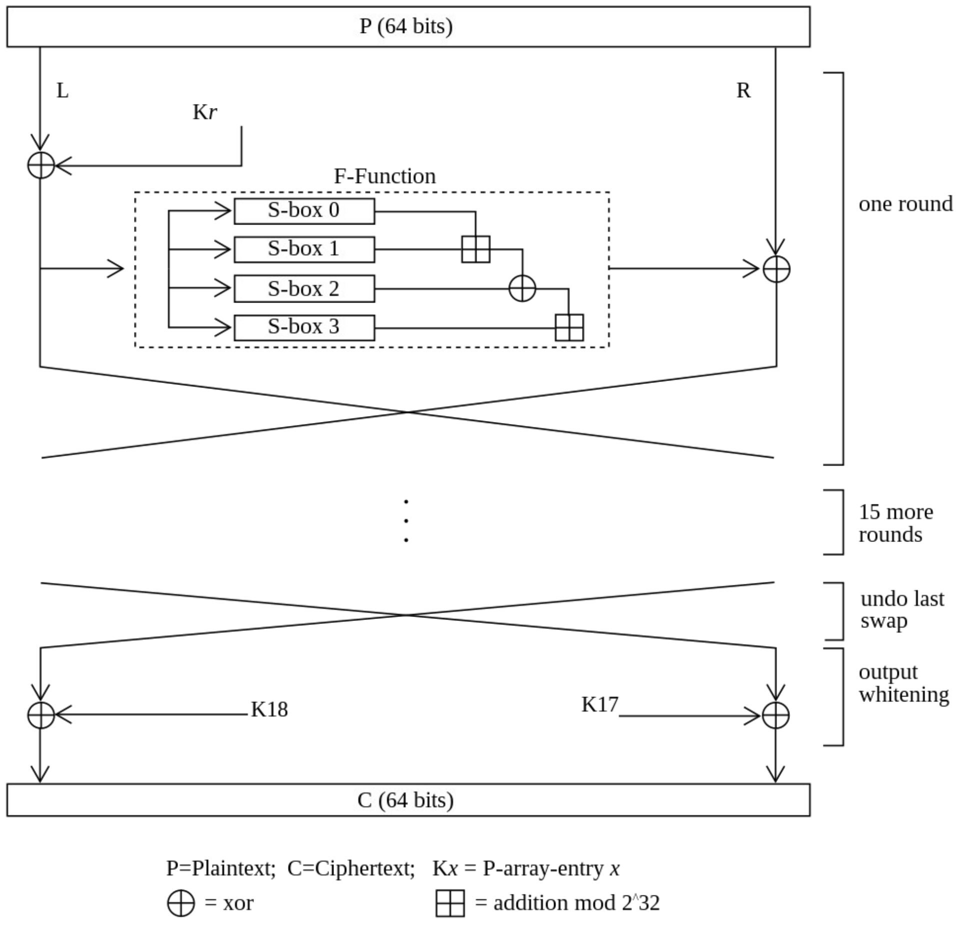
```c
#include <stdint.h>

void encrypt (uint32_t v[2], const uint32_t k[4]) {
    uint32_t v0=v[0], v1=v[1], sum=0, i;           /* set up */
    uint32_t delta=0x9E3779B9;                     /* a key schedule constant */
    uint32_t k0=k[0], k1=k[1], k2=k[2], k3=k[3];   /* cache key */
    for (i=0; i<32; i++) {                         /* basic cycle start */
        sum += delta;
        v0 += ((v1<<4) + k0) ^ (v1 + sum) ^ ((v1>>5) + k1);
        v1 += ((v0<<4) + k2) ^ (v0 + sum) ^ ((v0>>5) + k3);
    }                                              /* end cycle */
    v[0]=v0; v[1]=v1;
}

void decrypt (uint32_t v[2], const uint32_t k[4]) {
    uint32_t v0=v[0], v1=v[1], sum=0xC6EF3720, i;  /* set up; sum is (delta << 5) & 0xFFFFFFFF */
    uint32_t delta=0x9E3779B9;                     /* a key schedule constant */
    uint32_t k0=k[0], k1=k[1], k2=k[2], k3=k[3];   /* cache key */
    for (i=0; i<32; i++) {                         /* basic cycle start */
        v1 -= ((v0<<4) + k2) ^ (v0 + sum) ^ ((v0>>5) + k3);
        v0 -= ((v1<<4) + k0) ^ (v1 + sum) ^ ((v1>>5) + k1);
        sum -= delta;
    }                                              /* end cycle */
    v[0]=v0; v[1]=v1;
}
```

Another alternative to AES: Blowfish (Twofish was in the AES competition)

https://en.wikipedia.org/wiki/Blowfish_(cipher)



AES S-box requirements

- Can’t pull it out of our &%# like the NSA did for DES
- Should have good nonlinear properties
  - Better nonlinearity means fewer rounds
- Should be reversible
  - Don’t want to use a Feistel structure for performance reasons

https://en.wikipedia.org/wiki/Kaisa_Nyberg

The MixColumns operation also uses Galois fields (but is a linear function)...

```
private byte GMul(byte a, byte b) { // Galois Field (256) Multiplication of two Bytes
    byte p = 0;

    for (int counter = 0; counter < 8; counter++) {
        if ((b & 1) != 0) {
            p ^= a;
        }

        bool hi_bit_set = (a & 0x80) != 0;
        a <<= 1;
        if (hi_bit_set) {
            a ^= 0x1B; /* x^8 + x^4 + x^3 + x + 1 */
        }
        b >>= 1;
    }

    return p;
}

private void MixColumns() { // 's' is the main State matrix, 'ss' is a temp matrix of the same dimensions as 's'.
    Array.Clear(ss, 0, ss.Length);

    for (int c = 0; c < 4; c++) {
        ss[0, c] = (byte)(GMul(0x02, s[0, c]) ^ GMul(0x03, s[1, c]) ^ s[2, c] ^ s[3, c]);
        ss[1, c] = (byte)(s[0, c] ^ GMul(0x02, s[1, c]) ^ GMul(0x03, s[2, c]) ^ s[3,c]);
        ss[2, c] = (byte)(s[0, c] ^ s[1, c] ^ GMul(0x02, s[2, c]) ^ GMul(0x03, s[3, c]));
        ss[3, c] = (byte)(GMul(0x03, s[0,c]) ^ s[1, c] ^ s[2, c] ^ GMul(0x02, s[3, c]));
    }

    ss.CopyTo(s, 0);
}
```

https://en.wikipedia.org/wiki/Rijndael_MixColumns

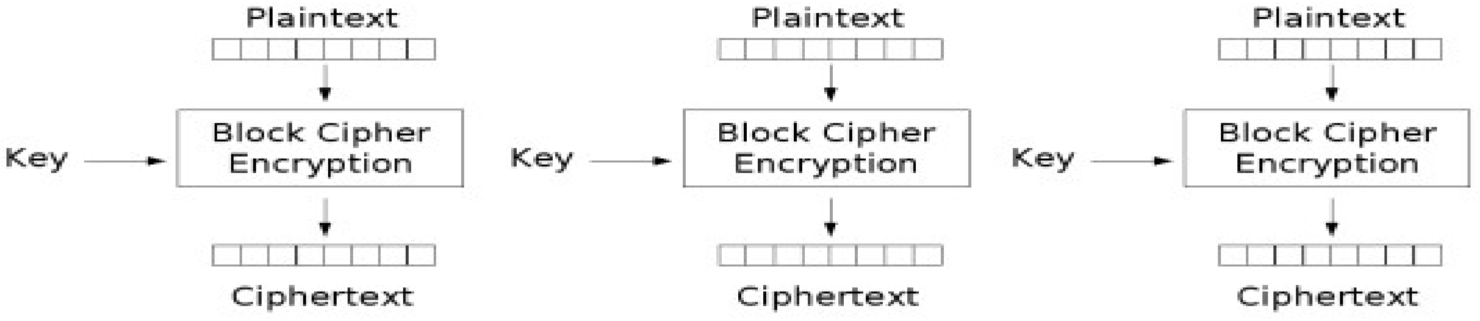
AES

- 128-bit blocks, 128-, 192-, or 256-bit keys
  - 10, 12, or 14 rounds respectively
- No less secure than the other candidates, but better performance…
  - In hardware *and* software
    - Different word sizes (8, 16, 32, 64)
  - With or without specialized hardware support
    - *E.g.*, Gallois Fields on Blackfin DSPs
    - *E.g.*, AES special instruction set on Intel chips

Cipher modes

- ECB, CBC discussed in the next slides
- Also Counter Mode, Galois Counter Mode, Cipher Feedback, Output Feedback, more...
  - Parallelization, message authentication, and other features
  - Can make stream ciphers out of block ciphers

Electronic Codebook (ECB)



Electronic Codebook (ECB) mode encryption

Image stolen from Wikipedia

Cipher Block Chaining (CBC)

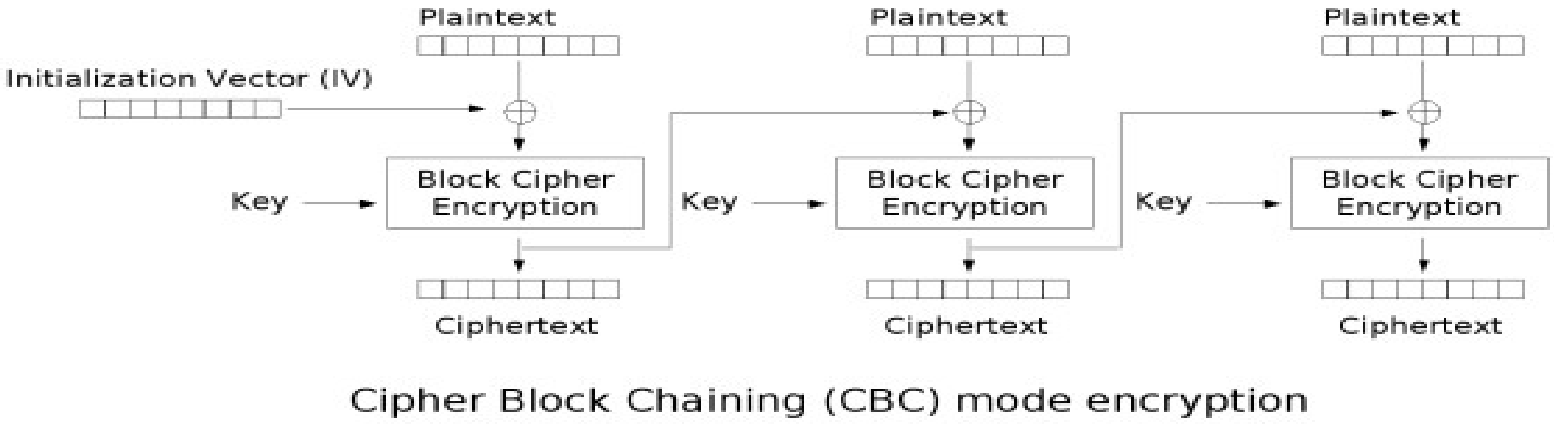
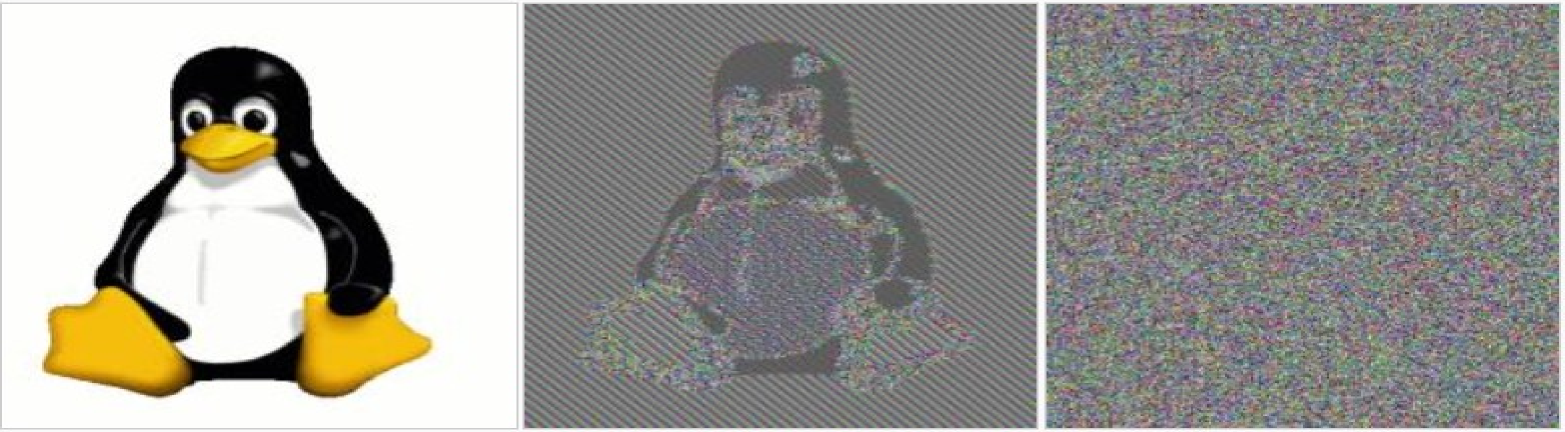


Image stolen from Wikipedia

ECB is generally bad

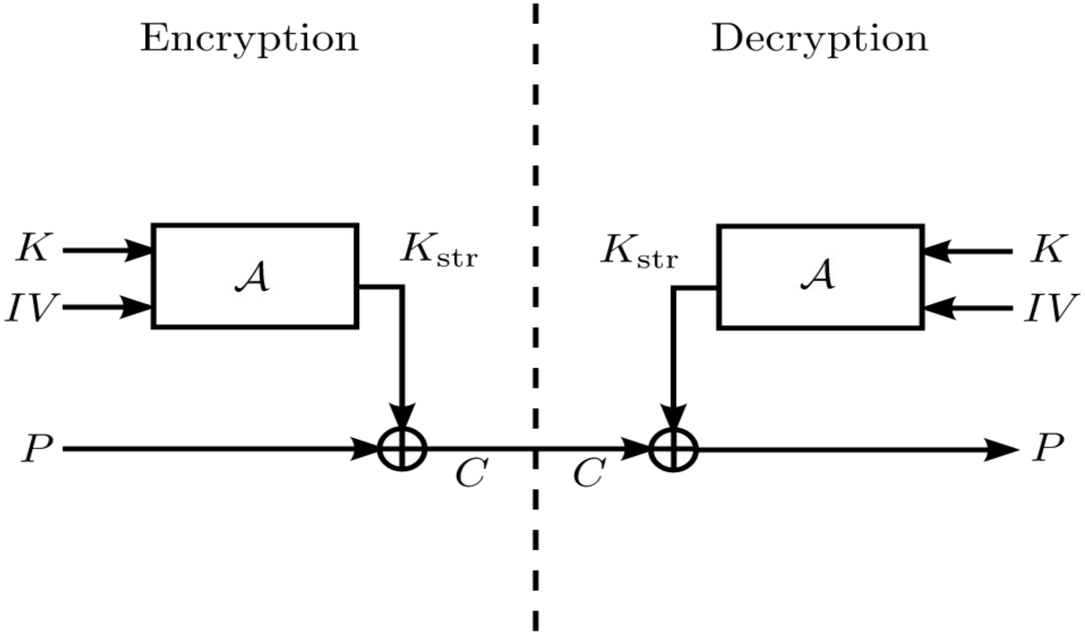


Original image | Encrypted using ECB mode | Modes other than ECB result in pseudo-randomness

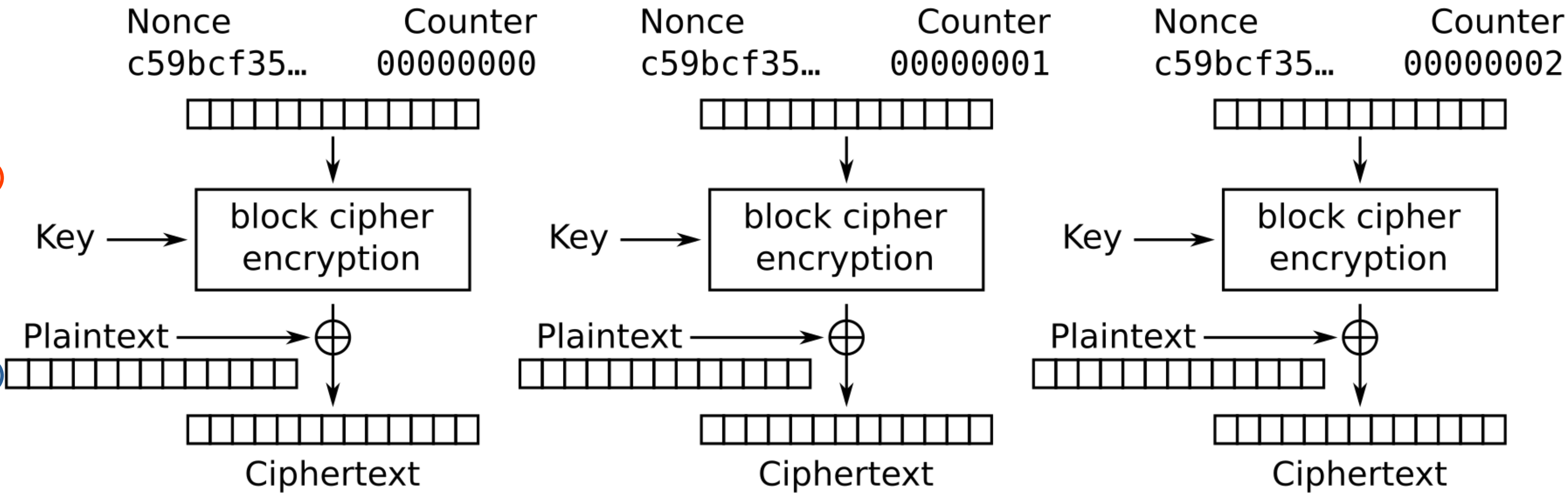
The image on the right is how the image might appear encrypted with CBC, CTR or any of the other more secure modes—indistinguishable from random noise. Note that the random appearance of the image on the right does not ensure that the image has been securely encrypted; many kinds of insecure encryption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

Image stolen from Wikipedia

Stream ciphers can be built out of block ciphers

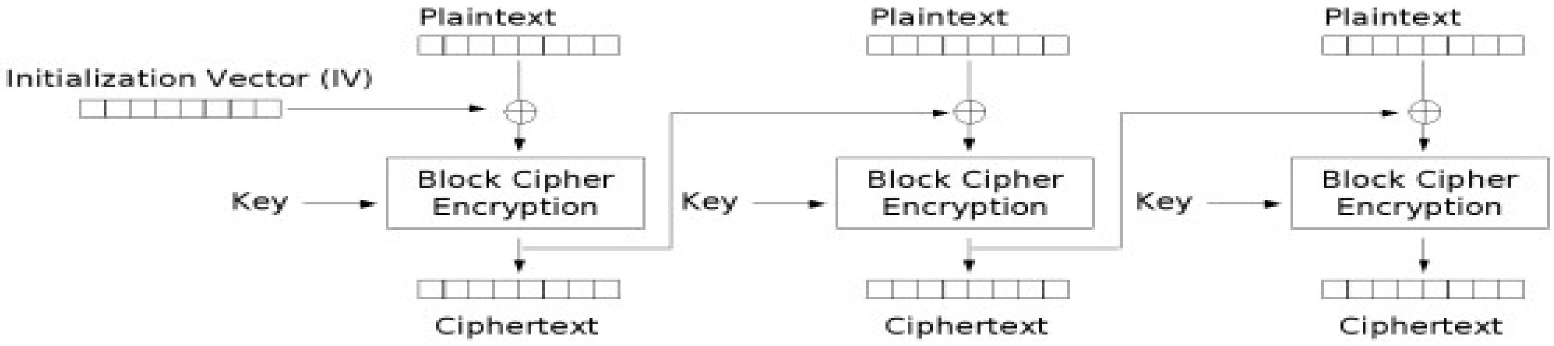


Stream ciphers can be built out of block ciphers



Counter (CTR) mode encryption

CBC padding oracle attacks...



Cipher Block Chaining (CBC) mode encryption

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

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)

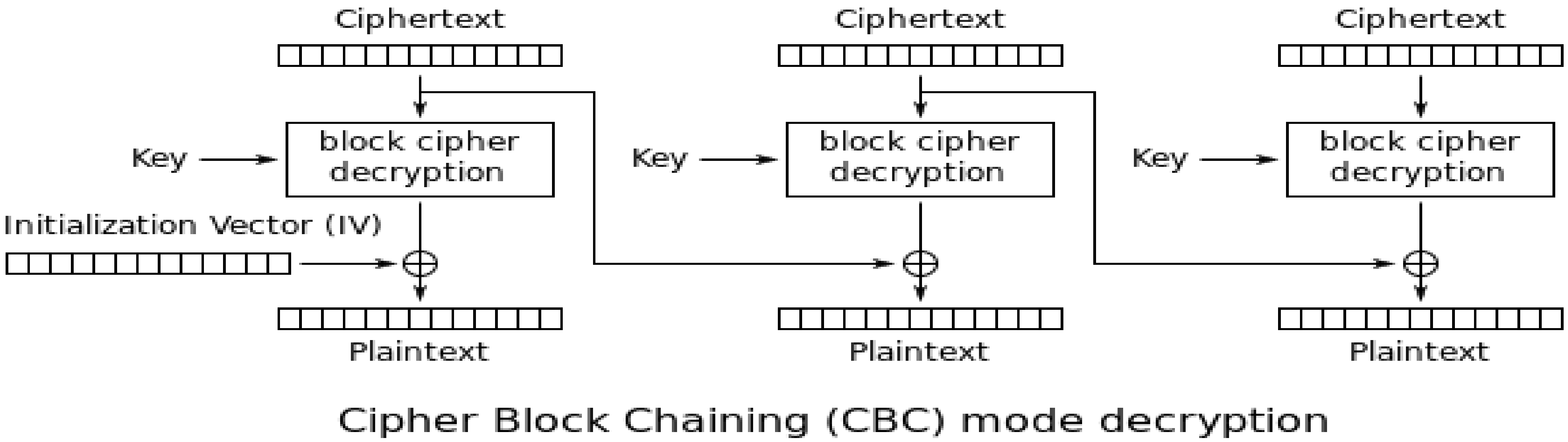
PKCS#7 padding

- 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)

															01
														02	02
													03	03	03
												04	04	04	04
											05	05	05	05	05
										06	06	06	06	06	06
									07	07	07	07	07	07	07
								08	08	08	08	08	08	08	08
							09	09	09	09	09	09	09	09	09
						0A	0A	0A	0A	0A	0A	0A	0A	0A	0A
					0B	0B	0B	0B	0B	0B	0B	0B	0B	0B	0B
				0C	0C	0C	0C	0C	0C	0C	0C	0C	0C	0C	0C
			0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D
		0E	0E	0E	0E	0E	0E	0E	0E	0E	0E	0E	0E	0E	0E
	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F	0F
10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10

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)


Ciphertext
Ciphertext
Ciphertext
Key
block cipher decryption
Key
block cipher decryption
Key
block cipher decryption
Initialization Vector (IV)
Plaintext
Plaintext
Plaintext
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	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

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	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

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	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

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	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

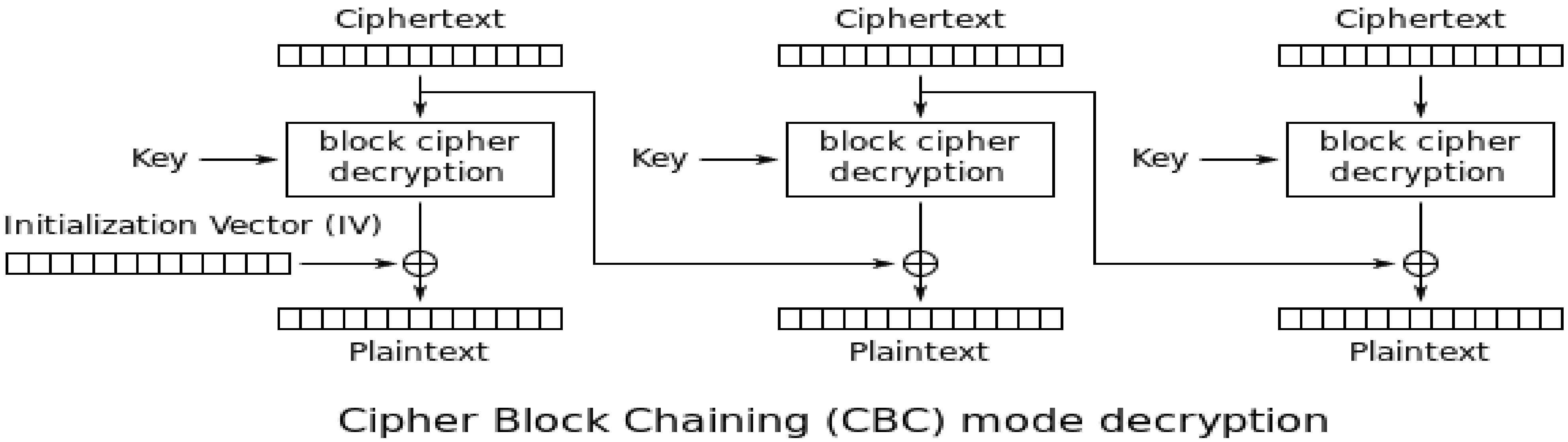
Try every value of this byte from 00 to FF

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

N	u	m	b	l	k	s	:	1	K	e	y	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

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



By WhiteTimberwolf (SVG version) - PNG version, Public Domain, https://commons.wikimedia.org/w/index.php?curid=26434095

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 XOR 02 == 01, so... Q == 01 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	I	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	[illegible]	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 XOR 02 = 03

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	01
98	CC	BE	01	FF	26	39	97	[illegible]	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 XOR 02 = 03

Hold this at 01

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	01
98	CC	BE	01	FF	26	39	97	[illegible]	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/

Cryptography Engineering by Ferguson *et al.*