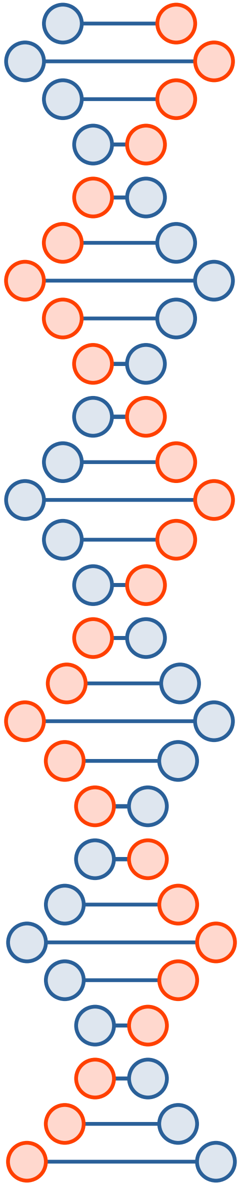
AES and cipherblock chaining modes

CSE 548 Spring 2024

jedimaestro@asu.edu

Why study AES?

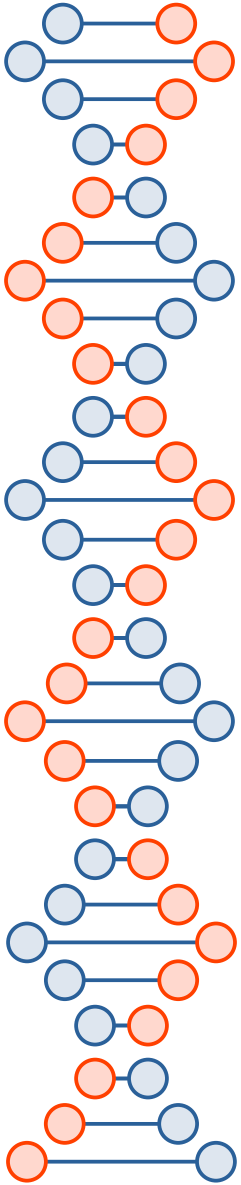
- It's a good demonstration of principles we'll be talking about (*e.g.*, confusion and diffusion).
- It's the workhorse of the majority of network crypto, *e.g.*, TLS
- It's easy to see that AES is just substitution, permutation, and XOR



Substitution

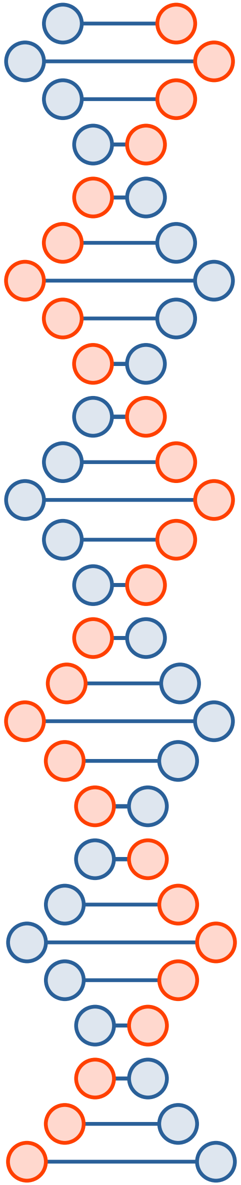
HELLO WORLD

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

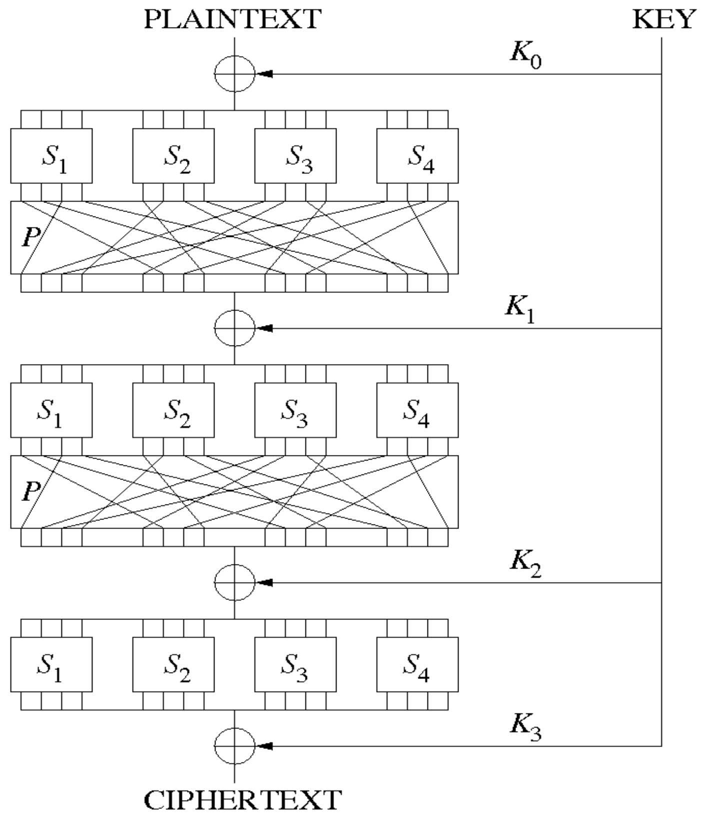
$$
\begin{aligned}
& 00101010_b \\
\oplus & 10000110_b \\
= & 10101100_b
\end{aligned}
$$

Symmetric encryption over time (review)

- 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

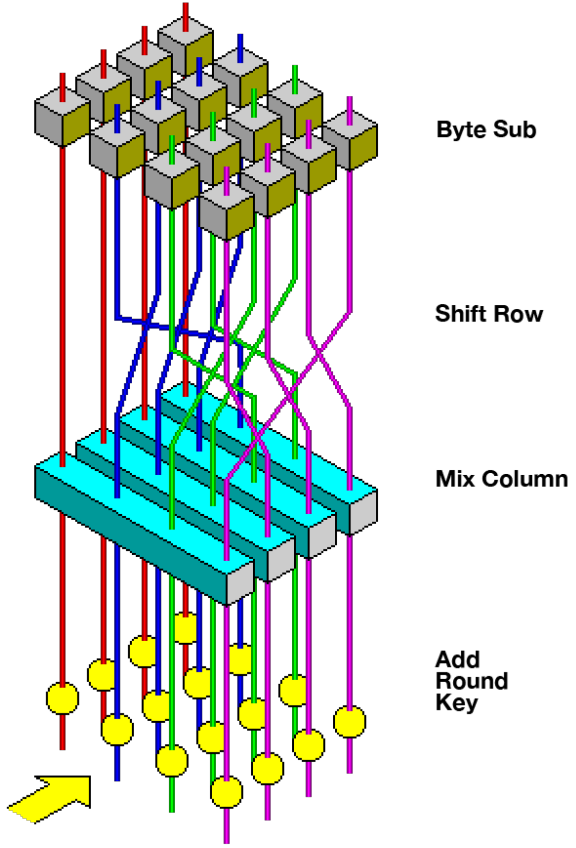
Substitution Permutation Network

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)



Finite fields

- In computer science, we like to pack things into the natural numbers between 0 and $2^n - 1$
- Would also be nice if things wrapped around and had other nice mathematical properties so we could pretend we didn't have this limitation
- *Really* nice if we don't waste any of the state space, *i.e.*, we use everything from 0 through $2^n - 1$

https://en.wikipedia.org/wiki/%C3%89variste_Galois

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

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

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

$$\frac{a}{q}x^2 + \frac{bq + ap}{q^2}x + \frac{cq^2 + bpq + ap^2}{q^3}$$

What is a field?

- “In mathematics, a field is a set on which addition, subtraction, multiplication, and division are defined and behave as the corresponding operations on rational and real numbers do.”

  --Wikipedia

- In cryptography, we often want to “undo things” or get the same result two different ways
  - Math!
- On digital computers the math you learned in grade school is not good enough
  - Suppose we want to multiply by a plaintext, and the plaintext is 3. Great!
  - Now the decryption needs the inverse operation. Crap!
  - 1/3 is not easy to deal with (not even in floating point or fixed point)

Field

- Commutative

  $a + b = b + a$

  $a * b = b * a$

- Associative

  $(a + b) + c = a + (b + c)$

  $(a * b) * c = a * (b * c)$

- Identity

  $0 != 1, a + 0 = a, a * 1 = a$

- Inverse

  $a + -a = 0$

  $a * a^{-1} = 1$

- Distributive

  $a * (b + c) = (a * b) + (a * c)$

Integers mod 100

- Commutative? Associative? Identity?
- Inverse?

Integers mod 100

- Commutative? Associative? Identity?
- ~~Inverse?~~
  - Sometimes there is one, e.g., 3 and 67 (201 % 100 = 1)
  - Sometimes not, e.g., 5
  - Integers mod 100 is **not** a finite field!

Integers mod 101

- Commutative? Associative? Identity?
- Inverse?
  - Every number $0 < i < 101$ has a multiplicative inverse
    - Co-prime to 101, because 101 is prime
  - Integers mod 101 **is** a finite field!
    - True of any prime number
    - In general p^k where p is prime and k is positive integer

GF(2)

- Want to define a field over 2^k possibilities for a k-bit number
- 2 is prime, all other powers of 2 are not
  - Need to use irreducible polynomials

https://jedcrandall.github.io/courses/cse548spring2024/miniaesspec.pdf

Published in Cryptologia, XXVI (4), 2002.

Mini Advanced Encryption Standard (Mini-AES): A Testbed for Cryptanalysis Students

Raphael Chung-Wei **Phan**

2.1 The Finite Field $GF(2^4)$

The nibbles of Mini-AES can be thought of as elements in the finite field $GF(2^4)$. Finite fields have the special property that operations (+,–, × and ÷) on the field elements always cause the result to be also in the field. Consider a nibble $n = (n_3, n_2, n_1, n_0)$ where $n_i \in \{0,1\}$. Then, this nibble can be represented as a polynomial with binary coefficients i.e having values in the set {0,1}:

$$n = n_3 x^3 + n_2 x^2 + n_1 x + n_0$$

Example 1

Given a nibble, n = 1011, then this can be represented as

$$n = 1 x^3 + 0 x^2 + 1 x + 1 = x^3 + x + 1$$

Note that when an element of $GF(2^4)$ is represented in polynomial form, the resulting polynomial would have a degree of at most 3.

2.2 Addition in GF(2^4)

When we represent elements of GF(2^4) as polynomials with coefficients in {0,1}, then addition of two such elements is simply addition of the coefficients of the two polynomials. Since the coefficients have values in {0,1}, then the addition of the coefficients is just modulo 2 addition or exclusive-OR denoted by the symbol ⊕. Hence, for the rest of this paper, the symbols + and ⊕ are used interchangeably to denote addition of two elements in GF(2^4).

Example 2

Given two nibbles, n = 1011 and m = 0111, then the sum, n + m = 1011 + 0111 = 1100 or in polynomial notation:

$$n + m = (x^3 + x + 1) + (x^2 + x + 1) = x^3 + x^2$$

2.3 Multiplication in GF(2^4)

Multiplication of two elements of GF(2^4) can be done by simply multiplying the two polynomials. However, the product would be a polynomial with a degree possibly higher than 3.

Example 3

Given two nibbles, n = 1011 and m = 0111, then the product is:

$$(x^3 + x + 1)(x^2 + x + 1) = x^5 + x^4 + x^3 + x^3 + x^2 + x + x^2 + x + 1$$
$$= x^5 + x^4 + 1$$

In order to ensure that the result of the multiplication is still within the field GF(2^4), it must be reduced by division with an irreducible polynomial of degree 4, the remainder of which will be taken as the final result. An irreducible polynomial is analogous to a prime number in arithmetic, and as such a polynomial is irreducible if it has no divisors other than 1 and itself. There are many such irreducible polynomials, but for Mini-AES, it is chosen to be:

$$m(x) = x^4 + x + 1$$

Example 4

Given two nibbles, n = 1011 and m = 0111, then the final result after multiplication in $GF(2^4)$, called the 'product of $n \times m$ modulo m(x)' and denoted as $\otimes$, is:

$$
\begin{aligned}
(x^3 + x + 1) \otimes (x^2 + x + 1) \quad &= x^5 + x^4 + 1 \text{ modulo } x^4 + x + 1 \\
&= x^2
\end{aligned}
$$

This is because:

$$
\begin{array}{r|l l}
 & \quad x + 1 & \text{(quotient)} \\
x^4 + x + 1 & x^5 + x^4 + 1 & \\
 & + \; x^5 + x^2 + x & \\
\hline
 & \quad x^4 + x^2 + x + 1 & \\
 & + \quad x^4 + \quad\; x + 1 & \\
\hline
 & \quad x^2 & \text{(remainder)}
\end{array}
$$

Note that since the coefficients of the polynomials are in {0,1}, then addition is simply exclusive-OR and hence subtraction is also exclusive-OR since exclusive-OR is its own inverse.

⊗	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

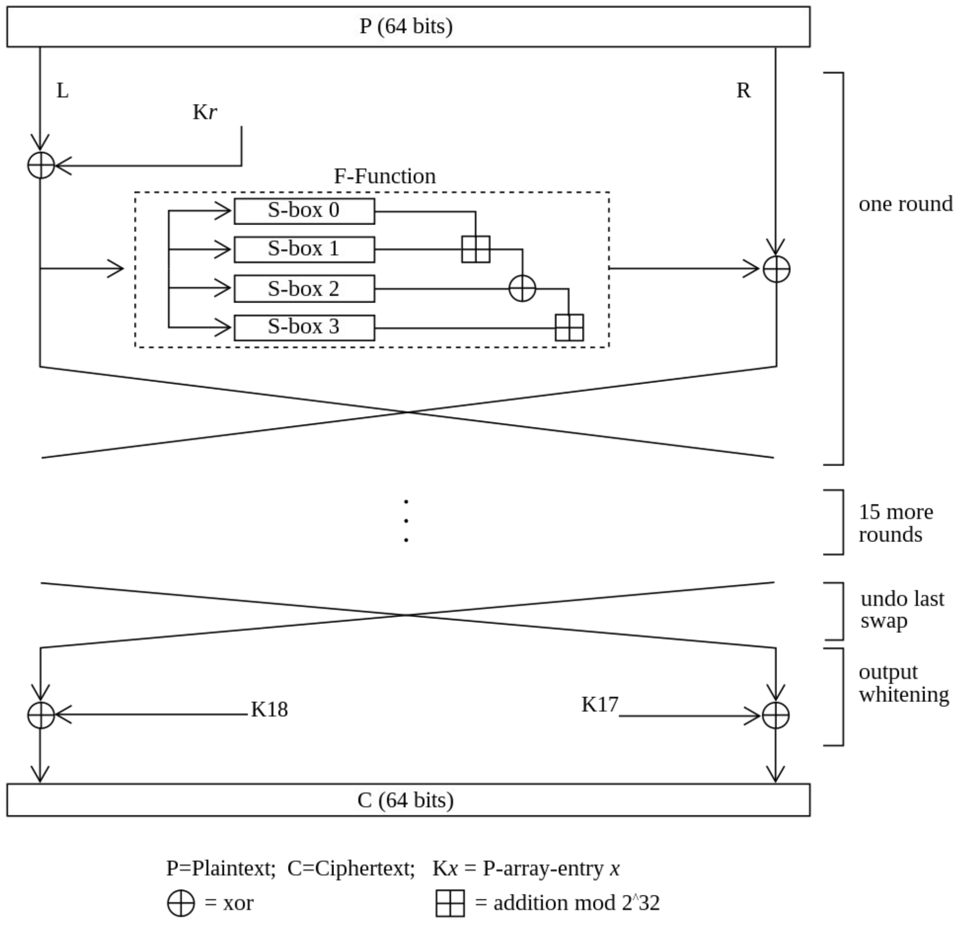
```c
#include <stdint.h>

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)

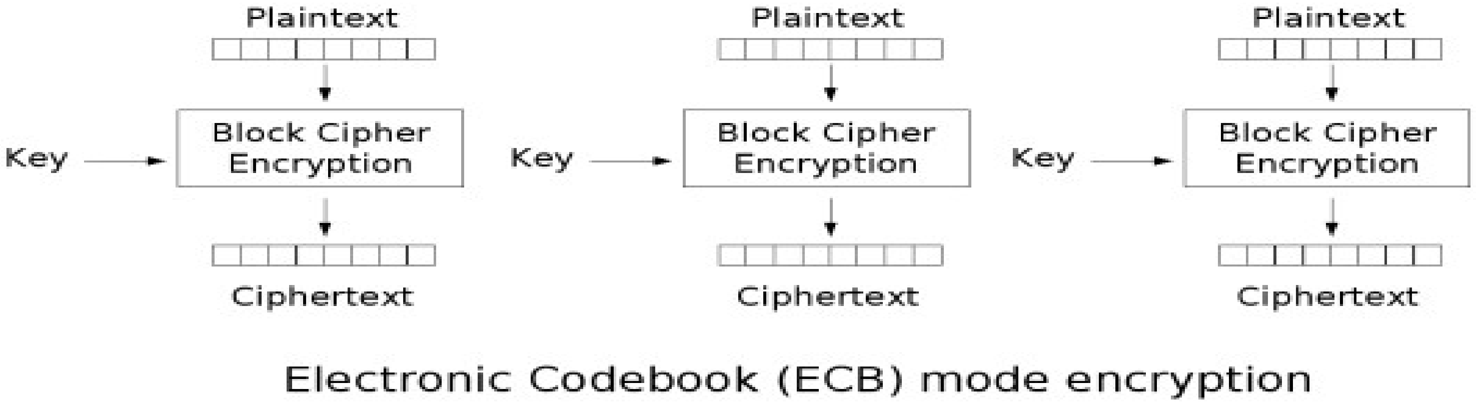
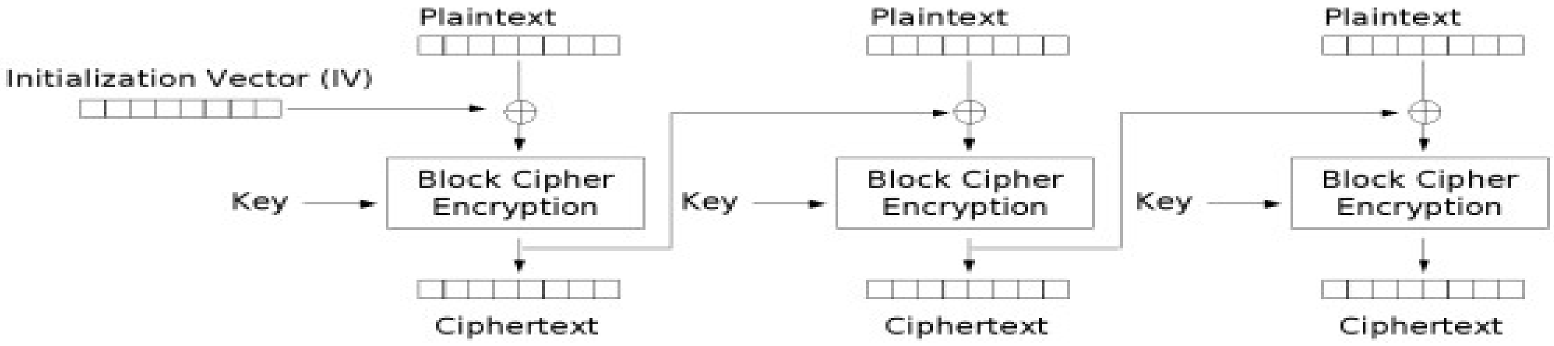


Image stolen from Wikipedia

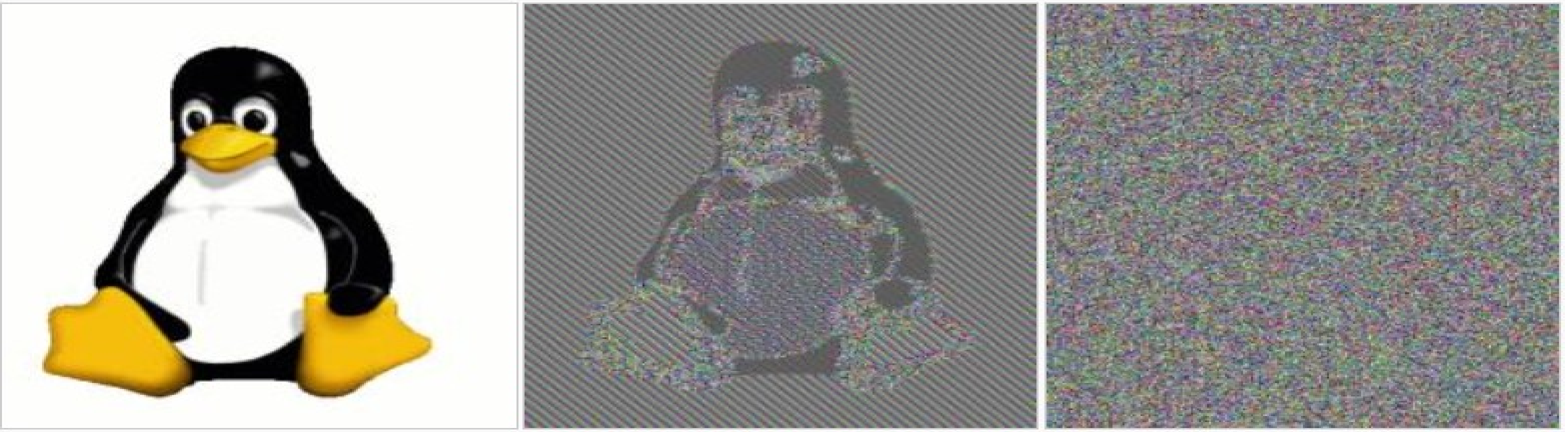
Cipher Block Chaining (CBC)



Cipher Block Chaining (CBC) mode encryption

Image stolen from Wikipedia

ECB is generally bad

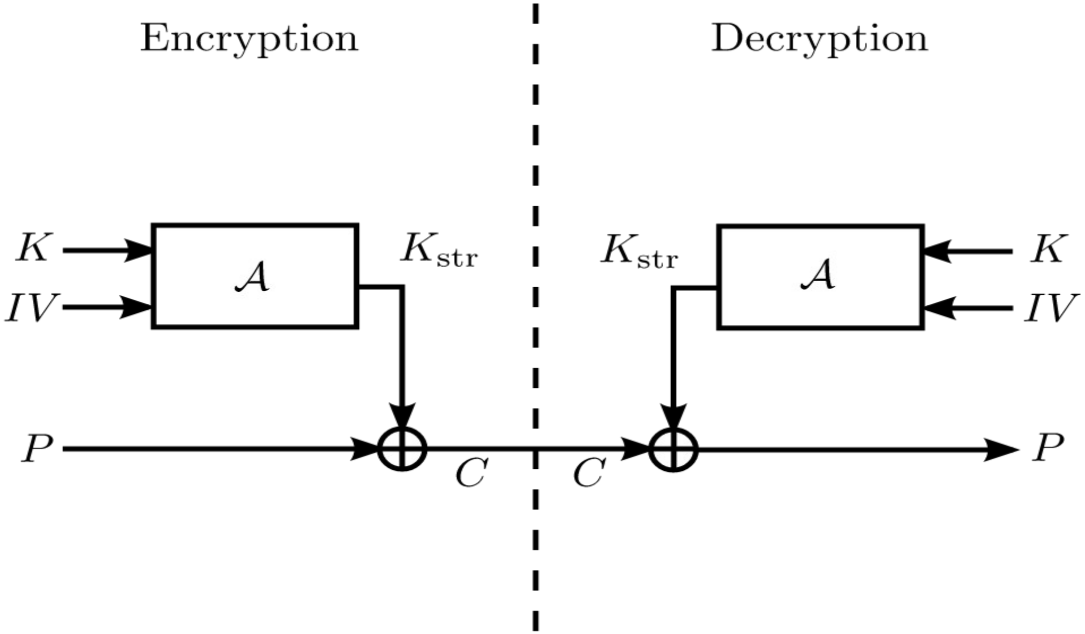


Original image | Encrypted using ECB mode | Modes other than ECB result in pseudo-randomness

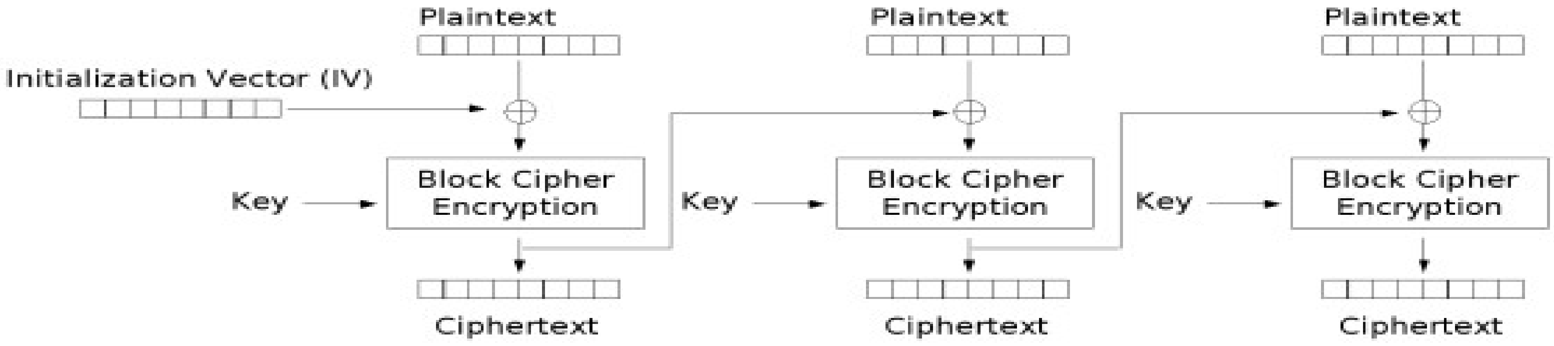
The image on the right is how the image might appear encrypted with CBC, CTR or any of the other more secure modes—indistinguishable from random noise. Note that the random appearance of the image on the right does not ensure that the image has been securely encrypted; many kinds of insecure encryption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

Image stolen from Wikipedia

Stream cipher (preview)

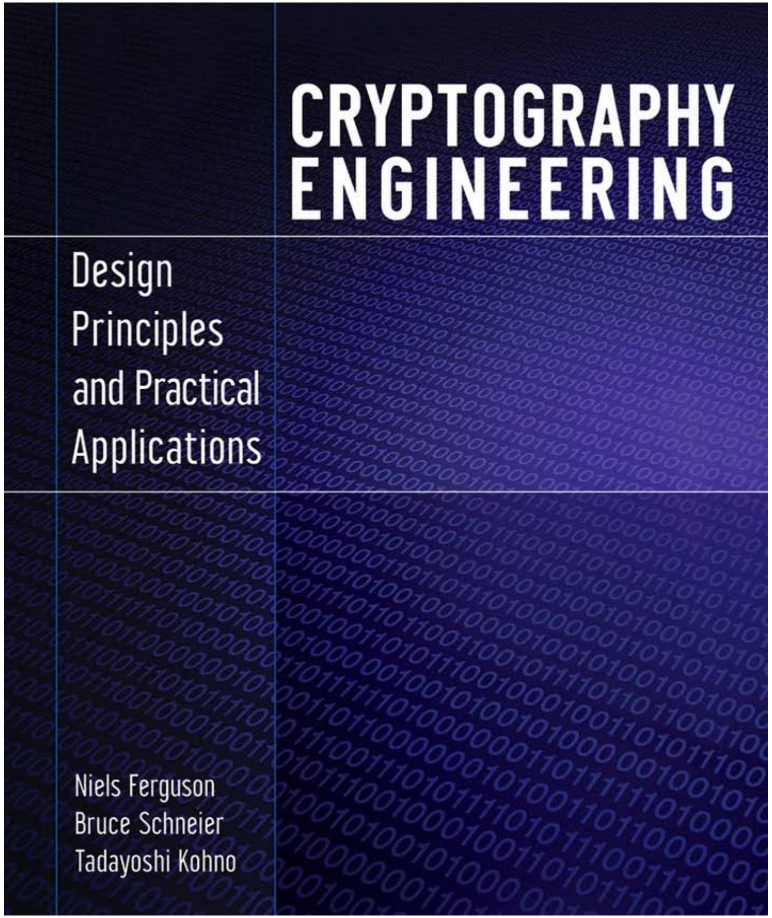


CBC padding oracle attacks (preview)



Cipher Block Chaining (CBC) mode encryption

Cryptography Engineering by Ferguson *et al.*



Next we want to formalize confusion and diffusion. For this and other things coming later in the semester, we need to study information theory...

A puzzle...

You have 12 coins, one is counterfeit. The counterfeit is either slightly heavier or slightly lighter, otherwise it's impossible to tell. You have a balance. Using the balance the fewest number of times, find the counterfeit coin.