Week 4 - Wednesday

COMP 4290

Last time

- What did we talk about last time?
- Secure encryption algorithms
- DES
- Started AES

Questions?

Project 1

Kyle Hinkle Presents

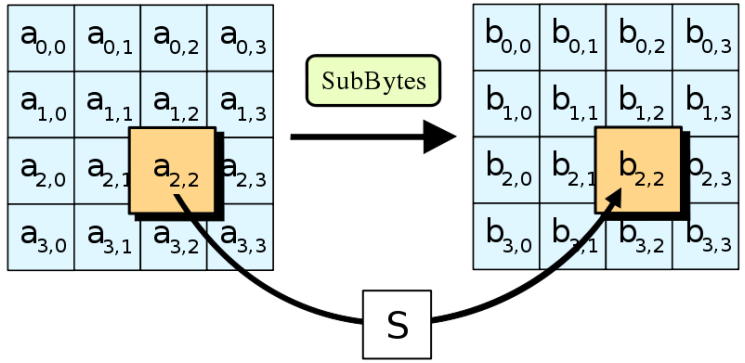
Back to AES

AES internals

- AES keeps an internal state of 128 bits in a 4×4 table of bytes
- There are four operations on the state:
  - Substitute bytes
  - Shift rows
  - Mix columns
  - Add round key

Substitute bytes

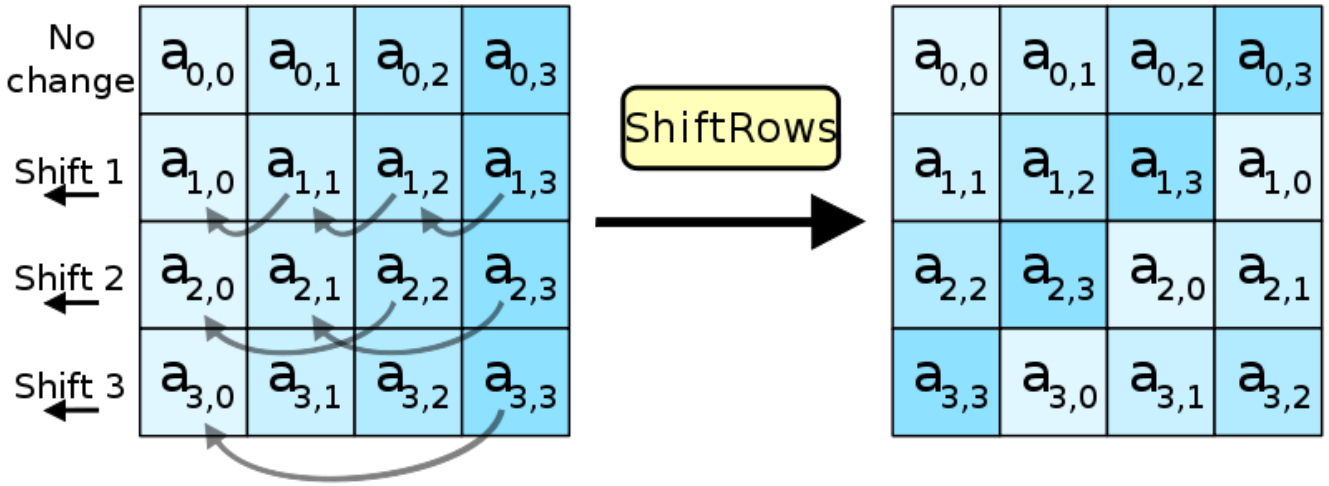
- Each byte is substituted for some other byte
- This operation is similar to the S-box from DES
- The substitution is based on the multiplicative inverse of the value in $GF(2^8)$
  - An algebraic structure is used instead of hand picking substitution value
  - o is used as its own multiplicative inverse
- To break up patterns, the result of finding the multiplicative inverse is XORed with the value 99



	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Shift rows

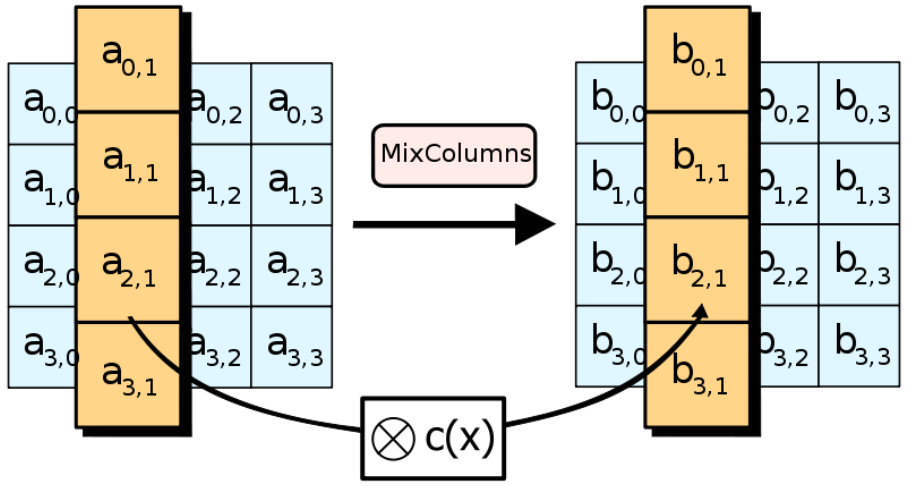
- For 128-bit blocks (those used in AES), the rows are shifted by a set amount
  - Row 1 is not shifted at all
  - Row 2 is shifted over by 1 byte
  - Row 3 is shifted over by 2 bytes
  - Row 4 is shifted over by 3 bytes
- Rijndael has slightly different shifts for larger block sizes



Mix columns

- Mixing the columns is the most confusing part
- An invertible linear transformation is applied to each column, diffusing its data along the column
- This transformation can be viewed as "multiplication" by the following matrix

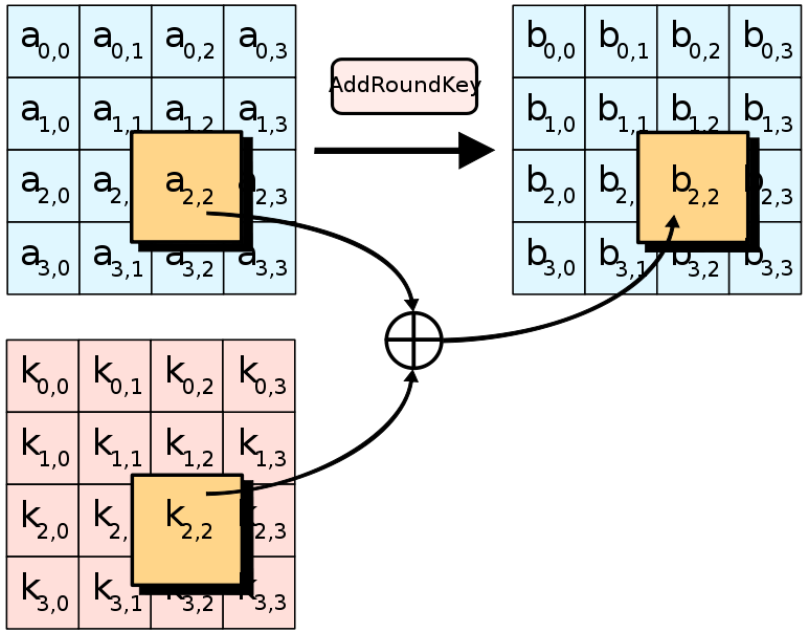
$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$



1 means don't change the value, 2 means left shift by one bit, and 3 means left shift by one bit and XOR with the original value

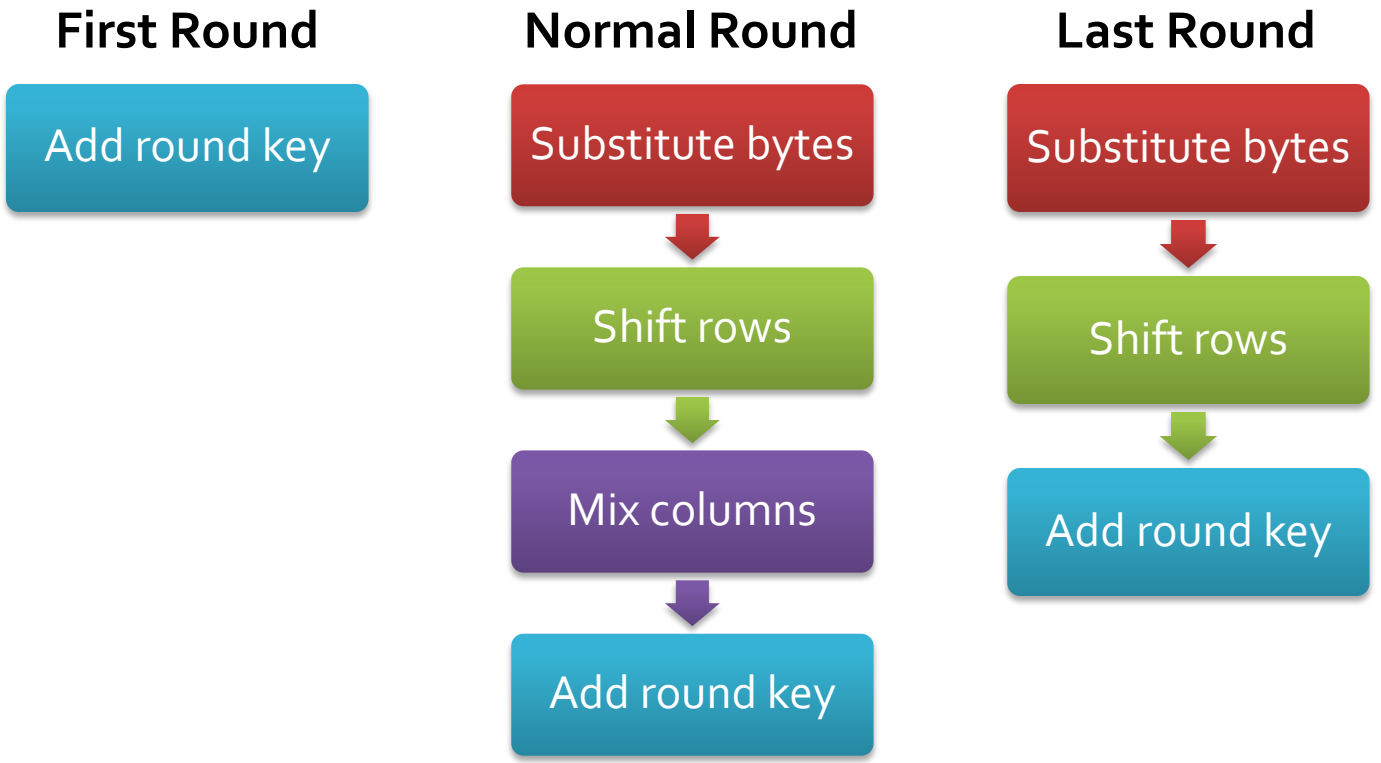
Add key

- XOR the current round key with the state
- This step is very simple, except that the key schedule that generates the round key from the overall key is complex



AES rounds

- AES supports key sizes of 128, 192, and 256 bits
  - Rijndael supports unlimited key size, in principle, as well as other block sizes
- 128-bit keys use 10 rounds, 192 use 12, and 256 use 14



AES pros and cons

- Strengths
  - Strong key size
  - Fast in hardware and software
  - Rich algebraic structure
  - Well-studied, open standard
- Weaknesses
  - Almost none
  - A few theoretical attacks exist on reduced-round numbers of AES
  - No practical attacks other than side-channel attacks

AES attacks

- No practical attacks exist on the full AES
- With reduced numbers of rounds and strong attack models, there are some theoretical attacks
  - CP = chosen plaintexts
  - RK-CP = related key chosen plaintexts

Rounds	Key Size	Data	Time	Year
6	All	2^{32} CP	2^{72}	1998
6	All	$6 \cdot 2^{32}$ CP	2^{44}	2000
7	192	$19 \cdot 2^{32}$ CP	2^{155}	2000
7	256	$21 \cdot 2^{32}$ CP	2^{172}	2000
7	All	$2^{128} - 2^{119}$ CP	2^{120}	2000
8	192	$2^{128} - 2^{119}$ CP	2^{188}	2000
8	256	$2^{128} - 2^{119}$ CP	2^{204}	2000
9	256	2^{85} RK-CP	2^{224}	2000
12	192	2^{123} RK-CP	2^{176}	2009
14	256	$2^{99.5}$ RK-CP	$2^{99.5}$	2009
10	128	2^{88} CP	$2^{126.1}$	2011

Side channel attacks

- Attacks that rely on timing, measuring cache, energy consumption, or other ways an implementation leaks data are called **side-channel attacks**
- Several practical side channel attacks for AES **do** exist
  - In 2005, Bernstein found a cache-timing attack that broke an OpenSSL implementation of AES using 200 million chosen plaintexts and a server that would give him precise timing data
  - Later in 2005, Osvik et al. found an attack that recovered a key after 800 encryptions in only 65 milliseconds, with software running on the target machine
  - In 2009, Saha et al. found an attack on hardware using differential fault analysis to recover a key with a complexity of 2^{32}
  - In 2010, Bangerter et al. found a cache-timing attack that required no knowledge of plaintexts or ciphertexts and could work in about 3 minutes after monitoring 100 encryptions
  - In 2016, Ashokkumar et al. found an attack that needs only 6-7 blocks of plaintext and ciphertext and runs in under a minute

AES vs. DES

	DES	AES
Date	1976	1999
Block size	64 bits	128 bits
Key length	56 bits	128, 192, 256 bits
Encryption primitives	Substitution, permutation	Substitution, shift, bit mixing
Cryptographic primitives	Confusion, diffusion	Confusion, diffusion
Design	Open	Open
Design rationale	Closed	Open
Selection process	Secret	Secret with public comment
Source	IBM with NSA help	Independent Belgians
Security	Broken if you've got the resources	No practical attacks yet

Public Key Cryptography

Symmetric key cryptography

- So far, we have talked about **symmetric** (or private) **key cryptography**
- In symmetric key cryptography, the same key is used for encryption and decryption
- The key is a **shared secret**
- This is perfect for sending messages between two parties who
  1. Trust each other
  2. Have shared a secret ahead of time

Public key cryptography

- Sometimes, we need something different
- We want a **public key** that anyone can use to encrypt a message to Alice
- Alice has a **private key** that can decrypt such a message
- The **public key** can only encrypt messages; it cannot be used to decrypt messages

Diffie and Hellman

- In 1976, Diffie and Hellman proposed the idea of a public key cryptosystem, one in which encryption and decryption keys were different
- They gave the following three conditions for such a system:
  1. It must be computationally easy to encipher or decipher a message given the appropriate key
  2. It must be computationally infeasible to derive the private key from the public key
  3. It must be computationally infeasible to determine the private key from a chosen plaintext attack

Public key history

- (Whitfield) Diffie and (Martin) Hellman "invented" public key cryptography in 1976
- However, James Ellis invented it in 1970, but his work was for a secret British government agency, classified until 1997
- Diffie and Hellman came up with the idea of a "trapdoor" function (computationally easy one way, hard the other)
- RSA, a practical algorithm published in 1978, made this idea workable
- Again, the system had been invented earlier by British intelligence
- The guys behind RSA made millions

Number Theory

Prime

- RSA depends in large part on the difficulty of factoring large composite numbers (particularly those that are a product of only 2 primes)
- Recall that an integer ***p*** is prime if
  - $p > 1$
  - ***p*** is not divisible by any positive integers other than 1 and itself

Fundamental theorem of arithmetic

- Any integer greater than 1 can be factored into a unique series of prime factors:
  - Example: $52 = 2^2 \cdot 13$

- Two integers ***a*** and ***b*** (greater than 1) are **relatively prime** or **coprime** if and only if ***a*** shares no prime factors with ***b***

Testing for primality

- How do we know if a number is prime?
- For small numbers, we can try to divide it by all integers less than or equal to its square root
- RSA-768 was successfully factored in December 2009 into 2 primes
  - One is:
    33478071698956898786044169848212690817704794983713768568912431
    388982883793 878002287614711652531743087737814467999489
- You can't test up to the square root of that in any reasonable time

Efficient primality testing

- In 2002, the AKS algorithm was published which demonstrated that it was possible to test to see if a number is prime
  - Deterministically
  - In time polynomial in the number of digits of the prime
- This algorithm is of theoretical interest, but it's too slow for testing the primality of RSA moduli

Rabin-Miller primality testing

- We won't get into the number theory behind this (yet)
- A Rabin-Miller primality test works as follows:
- Let n be the number you want to prove if it's prime or not
  - n must be odd, thus $n - 1$ is even
  - $(n - 1) = 2^s d$ where s and d are positive integers and d is odd
  - If n is prime, then for any integer $1 < a < n$, exactly one of the two is true:
    - $a^d \equiv 1 \pmod{n}$ or
    - $a^{2^r d} \equiv -1 \pmod{n}$, $1 \leq r < s$
  - Pick several a values, see if either of the two cases hold
  - If it ever doesn't, you know you have a composite

Rabin-Miller example

- What if we want to see if 221 is prime?
- $n - 1 = 220 = 2^2 \cdot 55$
- $s = 2$
- $d = 55$
- Attempt 1: Let $a = 174$
  - $a^{2^0 \cdot d} \bmod n = 174^{55} \bmod 221 = 47 \neq 1, n - 1$
  - $a^{2^1 \cdot d} \bmod n = 174^{110} \bmod 221 = 220 = n - 1$ **Check!**
- Attempt 2: Let $a = 137$
  - $a^{2^0 \cdot d} \bmod n = 137^{55} \bmod 221 = 188 \neq 1, n - 1$
  - $a^{2^1 \cdot d} \bmod n = 137^{110} \bmod 221 = 205 \neq n - 1$ **Oh no!**
- Every successful attempt means there is only a 25% chance that the number is composite
- So, after k attempts, there is a 4^{-k} chance that the number is composite

Ticket out the Door

Upcoming

Next time…

- RSA
- Public key management
- Spencer Wilson presents

Reminders

- **Office hours from 1:45-4 p.m. today are moved to 3-5 p.m.**
- Keep reading Sections 2.3 and 12.4
- Work on Project 1
  - Due this Friday