Discrete Structures Chapter 4.6 — Cryptography

Example 1 (Student Worksheet): Caesar Cipher, shift $k = 3$

Learning goals. Practice converting letters $\leftrightarrow$ numbers, computing $(p + k)$ mod 26, and translating back.

Alphabet convention (zero-based).

$$\mathrm{A} = 0,\ \mathrm{B} = 1,\ \ldots,\ \mathrm{Z} = 25$$

We work in $\mathbb{Z}_{26}$ (mod 26). Spaces and punctuation are carried through unchanged; we use uppercase.

Encryption rule. For plaintext number $p \in \{0, \ldots, 25\}$ and shift k, the ciphertext number is

$$c \equiv p + k \pmod{26}.$$

For this worksheet we use $\boxed{k = 3}$ (the classic "Caesar +3").

Fast tips (use 'em shamelessly):

- Add 3 quickly by doing *+1, +2, +3* as you scan, or use the wrap trick: adding 3 to $24, 25$ wraps to $1, 2$.
- Decrypting a +3 cipher is the same as *adding* -3, i.e., adding 23 mod 26.
- Common wrap cases: 24+3 $\to$ 1 (Y$\to$B), 25+3 $\to$ 2 (Z$\to$C).

Guided task. Encrypt the message:

MEET YOU IN THE PARK

Step 1 — Letters $\to$ numbers (A=0,...,Z=25). Fill the *plaintext numbers* p under each letter.

M	E	E	T	Y	O	U	I	N	T	H	E	P	A	R	K

(write numbers p here)

Step 2 — Add the shift $k = 3$ mod 26. Compute $c \equiv p + 3 \pmod{26}$ for each position and write the results:

Step 3 — Numbers $\to$ letters. Translate each c back to letters to form the ciphertext:

Neatness check. Your ciphertext should be readable in groups (keep the spaces from the original). If you decrypt with -3 you should land back on `MEET YOU IN THE PARK`.

Quick reference table (optional). If you like a visual:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Practice (still Caesar, but you drive):

P1. Encrypt (easy). Use $k = 5$ to encrypt:

`DOGS AND CATS`

Hint: D=3 so D↦3+5=8 ⇒ I. Keep spaces.

P2. Decrypt (easy). The ciphertext below was made with a $k = 5$ Caesar. Recover the plaintext.

`YMNX NX FQ YJXY`

Tip: Decrypt by adding -5 (or $+21$) mod 26.

P3. Crack the shift (harder). The message below is a Caesar cipher with *unknown* k:

`L ORYH PDWKP`

Clues: Try common words; guess that "`PDWKP`" might be "`MATH?`" or "`MATHS?`". Also, a one-letter word is often `A` or `I`. Determine k and decrypt.

Reflection. In one sentence: why does "mod 26" make the Caesar cipher *wrap* from Z back to A?

Example 1: Caesar Cipher ($k = 3$)

Question. Encrypt the message `MEET YOU IN THE PARK` using the Caesar cipher with shift $k = 3$.

Step 1 — Letters → numbers.

We use zero-based numbering: A=0, B=1, . . . , Z=25.

$$\text{MEET YOU IN THE PARK} \quad \Rightarrow \quad 12, 4, 4, 19,\ 24, 14, 20,\ 8, 13,\ 19, 7, 4,\ 15, 0, 17, 10$$

Step 2 — Apply $f(p) = (p + 3) \bmod 26$.

Add 3 to each number, wrapping around if the result exceeds 25:

$$(12 + 3) = 15,\ (4 + 3) = 7,\ (4 + 3) = 7,\ (19 + 3) = 22,$$

$$(24 + 3) = 27 \equiv 1,\ (14 + 3) = 17,\ (20 + 3) = 23,$$

$$(8 + 3) = 11,\ (13 + 3) = 16,\ (19 + 3) = 22,\ (7 + 3) = 10,\ (4 + 3) = 7,$$

$$(15 + 3) = 18,\ (0 + 3) = 3,\ (17 + 3) = 20,\ (10 + 3) = 13.$$

Step 3 — Numbers → letters.

Convert the ciphertext numbers back to letters:

$$15, 7, 7, 22,\ 1, 17, 23,\ 11, 16,\ 22, 10, 7,\ 18, 3, 20, 13$$

$$\Rightarrow \text{PHHW BRX LQ WKH SDUN}$$

Final Answer. The encrypted message is:

$$\boxed{\texttt{PHHW BRX LQ WKH SDUN}}$$

(Translation: "MEET YOU IN THE PARK" shifted +3.)

Quick Reflection. The Caesar cipher uses modular arithmetic in $\mathbb{Z}_{26}$ so letters "wrap around" after Z. The function $f(p) = (p + k) \bmod 26$ keeps all results in 0–25.

—

Practice Solutions

P1 — Encrypt (easy). Use $k = 5$ to encrypt: `DOGS AND CATS`.

Step 1 — Convert to numbers:

$$3, 14, 6, 18,\ 0, 13, 3,\ 2, 0, 19, 18$$

Step 2 — Add 5 mod 26:

$$8, 19, 11, 23,\ 5, 18, 8,\ 7, 5, 24, 23$$

Step 3 — Back to letters:

$$\boxed{\texttt{ITLX FSI HFYX}}$$

—

P2 — Decrypt (easy). Decrypt `YMNX NX FQ YJXY` that was made with $k = 5$.

We reverse the shift: $c - 5 \pmod{26}$.

$$\text{Y} = 24 \to 19 = \text{T},\ \text{M} = 12 \to 7 = \text{H},\ \text{N} = 13 \to 8 = \text{I},\ \text{X} = 23 \to 18 = \text{S}$$

$$\Rightarrow \boxed{\texttt{THIS IS AN TEST}}$$

So the message is "THIS IS AN TEST." (It should probably read "THIS IS A TEST.")

—

P3 — Crack the shift (harder). Ciphertext: `L ORYH PDWKP!`

Try guessing common English patterns.

`ORYH` looks like "LOVE," and the one-letter word "L" likely corresponds to "I."

That suggests a shift of $k = 3$ backward (since L $\to$ I is -3).

Decrypting with $k = 3$:

$$\texttt{L ORYH PDWKP!} \Rightarrow \boxed{\texttt{I LOVE MATH!}}$$

—

Summary of Key Takeaways

- The Caesar cipher is modular addition in $\mathbb{Z}_{26}$.
- Encryption: $E_k(p) = (p + k) \bmod 26$
- Decryption: $D_k(c) = (c - k) \bmod 26$
- If you can add or subtract mod 26, you can encrypt or decrypt.
- This cipher is historically important but easily broken by frequency analysis or brute force (26 possibilities).

Going Deeper. You can extend this same math to more complex ciphers:

$$f(p) = (a \cdot p + b) \bmod 26$$

where a must have a multiplicative inverse mod 26. This leads directly into the *Affine Cipher*—our next example.

Example 2 (Worksheet) — Shift Cipher with $k = 11$

Goal. Encrypt the message `STOP GLOBAL WARMING` using Caesar's shift cipher with $k = 11$.

Big idea (the "why"):

We model letters as numbers in $\mathbb{Z}_{26}$ so that a shift is just *modular addition*. This keeps us in the alphabet and gives the wrap-around from Z back to A.

$$\text{A} = 0,\ \text{B} = 1,\ \ldots,\ \text{Z} = 25 \qquad\qquad E_k(p) = (p + k) \bmod 26.$$

For this example, $k = 11$.

Step 1 — Normalize and map letters → numbers

We use uppercase and keep spaces. Convert each letter of `STOP GLOBAL WARMING` to its number:

$$\underbrace{\texttt{STOP}}_{18\ 19\ 14\ 15} \quad \underbrace{\texttt{GLOBAL}}_{6\ 11\ 14\ 1\ 0\ 11} \quad \underbrace{\texttt{WARMING}}_{22\ 0\ 17\ 12\ 8\ 13\ 6}.$$

Step 2 — Apply the shift $k = 11$ (add 11 mod 26)

Compute $c \equiv p + 11 \pmod{26}$ for each number. Do the wrap when you go past 25.

$$\begin{aligned} \texttt{STOP}: &\quad 18, 19, 14, 15 \mapsto 3, 4, 25, 0 \\ \texttt{GLOBAL}: &\quad 6, 11, 14, 1, 0, 11 \mapsto 17, 22, 25, 12, 11, 22 \\ \texttt{WARMING}: &\quad 22, 0, 17, 12, 8, 13, 6 \mapsto 7, 11, 2, 23, 19, 24, 17. \end{aligned}$$

Step 3 — Map numbers → letters and keep spaces

$$3, 4, 25, 0 \ \mid\ 17, 22, 25, 12, 11, 22 \ \mid\ 7, 11, 2, 23, 19, 24, 17 \quad \Rightarrow \quad \boxed{\texttt{DEZA RWZMLW HLCXTYR}}$$

Helpful tips & common pitfalls

- **A=0, not 1.** Off-by-one mistakes are the #1 bug.
- **Wrap cleanly:** if $p + k \geq 26$, subtract 26 (i.e., reduce mod 26).

- **Spaces/punctuation** pass through unchanged; only letters get shifted.
- **Decrypting** with $k = 11$ is the same as adding -11 (or $+15$) mod 26.

Practice (your turn!)

Problem A (easier). Encrypt with $k = 4$: `MATH IS FUN`

Why: smaller shift, shorter phrase—perfect confidence builder.

Problem B (similar). Decrypt with $k = 11$: `SPWWZ HZCWO`

Tip: subtract 11 mod 26 or add 15.

Problem C (harder). Unknown k. Decrypt the Caesar ciphertext: `P HT HA AOL WHYR`
Hints: a one-letter word is often `I` or `A`. The block `AOL` frequently shows up when "`THE`" is encrypted with $k = 7$.

Reflection. In one sentence: explain why modular arithmetic guarantees a valid letter after every shift.

Solutions for Example 2 Practice

Problem A (easier). Encrypt with $k = 4$: `MATH IS FUN`

Map to numbers (A=0):

$$\texttt{MATH IS FUN} \Rightarrow 12, 0, 19, 7,\ 8, 18,\ 5, 20, 13.$$

Add 4 mod 26:

$$12, 0, 19, 7 \mapsto 16, 4, 23, 11 \quad (\text{M} \to \text{Q},\ \text{A} \to \text{E},\ \ldots)$$

$$8, 18 \mapsto 12, 22 \qquad 5, 20, 13 \mapsto 9, 24, 17.$$

Back to letters:

$$16, 4, 23, 11,\ 12, 22,\ 9, 24, 17 \Rightarrow \boxed{\texttt{QEXL MW JYR}}.$$

Why it works: Every step is addition in $\mathbb{Z}_{26}$; wrap ensures letters stay in 0–25.

Problem B (similar). Decrypt with $k = 11$: `SPWWZ HZCWO`

Numbers for ciphertext:

$$\texttt{SPWWZ HZCWO} \Rightarrow 18, 15, 22, 22, 25,\ 7, 25, 2, 22, 14.$$

Subtract 11 (or add 15) mod 26:

$$18, 15, 22, 22, 25 \mapsto 7, 4, 11, 11, 14 \quad (\text{H,E,L,L,O})$$

$$7, 25, 2, 22, 14 \mapsto 22, 14, 17, 11, 3 \quad (\text{W,O,R,L,D}).$$

Plaintext: $\boxed{\texttt{HELLO WORLD}}$.

Problem C (harder). Unknown k: `P HT HA AOL WHYR`

Strategy (the why): Look for patterns. A one-letter word is probably `I` or `A`. Also, `AOL` famously appears when "`THE`" is shifted by $k = 7$ (since $19+7 = 26 \equiv 0 = \text{A}$, etc.).

Infer k: If `AOL` is `THE`, then the shift is $k = 7$.

Decrypt by subtracting 7:

$$\texttt{P} \mapsto \texttt{I}, \quad \texttt{HT} \mapsto \texttt{AM}, \quad \texttt{HA} \mapsto \texttt{AT}, \quad \texttt{AOL} \mapsto \texttt{THE}, \quad \texttt{WHYR} \mapsto \texttt{PARK}.$$

$\Rightarrow$ `I AM AT THE PARK`.

Key takeaways.

- Encryption: $E_k(p) = (p + k) \bmod 26$, Decryption: $D_k(c) = (c - k) \bmod 26$.
- Unknown k can be cracked with educated guesses ("`THE`", one-letter words) or brute force (only 26 options).
- Thinking in $\mathbb{Z}_{26}$ explains the wrap-around and keeps errors low.

Caesar Cipher Decryption

Student Worksheet

Understanding Decryption

Previously, we learned how to **encrypt** messages using the Caesar cipher. Now we'll learn to **decrypt** them—convert the secret message back to the original!

The key insight: **Decryption is the reverse of encryption.**

- **Encryption:** We shifted letters *forward* by k positions using $f(p) = (p + k) \bmod 26$
- **Decryption:** We shift letters *backward* by k positions using $f(p) = (p - k) \bmod 26$

Key Concept: Negative Numbers and Mod

When we subtract and get a negative number, we need to "wrap around" the other direction. For example, if we try to go back 7 from the letter E (position 4), we get $4 - 7 = -3$.

To handle this, we compute: $-3 \bmod 26 = 23$ (which is the letter X).

Quick trick: If you get a negative number, just add 26 to make it positive!

$$-3 + 26 = 23$$

Example 3: Worked Solution

Question: Decrypt the ciphertext message "LEWLYPLUJL PZ H NYLHA HSOHOLY" that was encrypted with the shift cipher with shift $k = 7$.

Solution:

Step 1: Convert letters to numbers

We use our standard A=0, B=1, C=2, ..., Z=25 system. Let's convert the ciphertext:

- **LEWLYPLUJL:** L=11, E=4, W=22, L=11, Y=24, P=15, L=11, U=20, J=9, L=11
- **PZ:** P=15, Z=25

- **H:** H=7
- **NYLHA:** N=13, Y=24, L=11, H=7, A=0
- **HSOHOLY:** A=0, L=11, H=7, O=14, L=11, Y=24

Our number sequence is:

$$11 \quad 4 \quad 22 \quad 11 \quad 24 \quad 15 \quad 11 \quad 20 \quad 9 \quad 11$$

$$15 \quad 25 \qquad 7 \qquad 13 \quad 24 \quad 11 \quad 7 \quad 0 \qquad \begin{matrix} 0 \\ 24 \end{matrix} \quad 11 \quad 7 \quad 9 \quad 14 \quad 11$$

Step 2: Apply the decryption function
We apply $f(p) = (p-7) \bmod 26$ to shift backward by 7. Let's work through each number:

$$\begin{aligned}
f(11) &= (11-7) \bmod 26 = 4 \bmod 26 = 4 \\
f(4) &= (4-7) \bmod 26 = -3 \bmod 26 = 23 \quad (\text{add } 26\text{: } -3+26=23) \\
f(22) &= (22-7) \bmod 26 = 15 \bmod 26 = 15 \\
f(11) &= (11-7) \bmod 26 = 4 \bmod 26 = 4 \\
f(24) &= (24-7) \bmod 26 = 17 \bmod 26 = 17 \\
f(15) &= (15-7) \bmod 26 = 8 \bmod 26 = 8 \\
f(11) &= (11-7) \bmod 26 = 4 \bmod 26 = 4 \\
f(20) &= (20-7) \bmod 26 = 13 \bmod 26 = 13 \\
f(9) &= (9-7) \bmod 26 = 2 \bmod 26 = 2 \\
f(11) &= (11-7) \bmod 26 = 4 \bmod 26 = 4
\end{aligned}$$

Continuing for the remaining letters:

$$\begin{aligned}
&f(15) = 8, \quad f(25) = 18, \quad f(7) = 0 \\
&f(13) = 6, \quad f(24) = 17, \quad f(11) = 4, \quad f(7) = 0, \quad f(0) = 19 \\
&f(0) = 19, \quad f(11) = 4, \quad f(7) = 0, \quad f(9) = 2, \quad f(14) = 7, \quad f(11) = 4, \quad f(24) = 17
\end{aligned}$$

Our decrypted numbers are:

$$4 \quad 23 \quad 15 \quad 4 \quad 17 \quad 8 \quad 4 \quad 13 \quad 2 \quad 4$$

$$8 \quad 18 \qquad 0 \qquad 6 \quad 17 \quad 4 \quad 0 \quad 19 \qquad 19 \quad 4 \quad 0 \quad 2 \quad 7 \quad 4 \quad 17$$

Pro Tip: Handling Negative Results

Whenever $(p - k)$ gives you a negative number:

1. Notice it's negative
2. Add 26 to make it positive
3. That's your answer!

Example: $(4 - 7) = -3$, so $-3 + 26 = 23$

Step 3: Convert numbers back to letters
Using A=0, B=1, . . . , Z=25:

- 4=E, 23=X, 15=P, 4=E, 17=R, 8=I, 4=E, 13=N, 2=C, 4=E
- 8=I, 18=S
- 0=A
- 6=G, 17=R, 4=E, 0=A, 19=T
- 19=T, 4=E, 0=A, 2=C, 7=H, 4=E, 17=R

Final Answer: The decrypted message is **EXPERIENCE IS A GREAT TEACHER**

Why This Works

If someone encrypted a message by shifting forward 7, we decrypt by shifting backward 7. It's like walking 7 steps forward, then 7 steps back—you end up where you started!

Practice Problems

Problem A (Easier Warm-up)

Decrypt the ciphertext "**FDW**" that was encrypted with shift $k = 3$.

Hint: This is a short message. Remember to subtract 3 from each letter's position. If you get negative numbers, add 26!

Problem B (Standard Practice)

Decrypt the ciphertext "**MJQQT BTWQI**" that was encrypted with shift $k = 5$.

Hint: You encrypted this message in the previous worksheet! Now decrypt it to get back the original message.

Problem C (Challenge)

Decrypt the ciphertext "**EJKKR ZRUOJ**" that was encrypted with shift $k = 5$.

Challenge: Some of these letters will give negative results when you subtract 5. Practice your wrapping-around skills!

Caesar Cipher Decryption

Teacher Solutions Manual

Problem A Solution: Decrypt "CAT" with shift $k = 3$

Step 1: Convert letters to numbers

$$\begin{aligned} \text{F} &= 5 \\ \text{D} &= 3 \\ \text{W} &= 22 \end{aligned}$$

Number sequence: 5 3 22

Step 2: Apply decryption function $f(p) = (p - 3) \bmod 26$

$$\begin{aligned} f(5) &= (5 - 3) \bmod 26 = 2 \bmod 26 = 2 \\ f(3) &= (3 - 3) \bmod 26 = 0 \bmod 26 = 0 \\ f(22) &= (22 - 3) \bmod 26 = 19 \bmod 26 = 19 \end{aligned}$$

Decrypted numbers: 2 0 19

Step 3: Convert back to letters

$$\begin{aligned} 2 &= \text{C} \\ 0 &= \text{A} \\ 19 &= \text{T} \end{aligned}$$

Answer: $\boxed{\textbf{CAT}}$

Teaching Note

This is the easiest problem because: (1) short message, (2) all results are positive (no negative numbers to handle), and (3) it's the reverse of Problem A from the encryption worksheet. Students can verify their answer by re-encrypting CAT with $k = 3$ to get FDW.

Problem B Solution: Decrypt "MJQQT BTWQI" with shift $k = 5$

Step 1: Convert letters to numbers

Breaking down by word:

- **MJQQT:** M=12, J=9, Q=16, Q=16, T=19
- **BTWQI:** B=1, T=19, W=22, Q=16, I=8

Number sequence:

$$12 \quad 9 \quad 16 \quad 16 \quad 19 \qquad 1 \quad 19 \quad 22 \quad 16 \quad 8$$

Step 2: Apply decryption function $f(p) = (p - 5) \bmod 26$

$$\begin{aligned}
f(12) &= (12 - 5) \bmod 26 = 7 \bmod 26 = 7 \\
f(9) &= (9 - 5) \bmod 26 = 4 \bmod 26 = 4 \\
f(16) &= (16 - 5) \bmod 26 = 11 \bmod 26 = 11 \\
f(16) &= (16 - 5) \bmod 26 = 11 \bmod 26 = 11 \\
f(19) &= (19 - 5) \bmod 26 = 14 \bmod 26 = 14 \\
f(1) &= (1 - 5) \bmod 26 = -4 \bmod 26 = 22 \quad (-4 + 26 = 22) \\
f(19) &= (19 - 5) \bmod 26 = 14 \bmod 26 = 14 \\
f(22) &= (22 - 5) \bmod 26 = 17 \bmod 26 = 17 \\
f(16) &= (16 - 5) \bmod 26 = 11 \bmod 26 = 11 \\
f(8) &= (8 - 5) \bmod 26 = 3 \bmod 26 = 3
\end{aligned}$$

Decrypted numbers:

$$7 \quad 4 \quad 11 \quad 11 \quad 14 \qquad 22 \quad 14 \quad 17 \quad 11 \quad 3$$

Step 3: Convert back to letters

- 7=H, 4=E, 11=L, 11=L, 14=O
- 22=W, 14=O, 17=R, 11=L, 3=D

Answer: **HELLO WORLD**

Teaching Note

This problem introduces negative numbers! When we decrypt B (position 1) with shift 5, we get: $1 - 5 = -4$.
To handle negative results in modular arithmetic: $-4 \bmod 26 = 22$
Students can calculate this by adding 26: $-4 + 26 = 22$, which corresponds to the letter W.
Connection: Students encrypted "HELLO WORLD" in the previous worksheet and got "MJQQT BTWQI". Now they're decrypting it back—reinforcing the inverse relationship between encryption and decryption.

Problem C Solution: Decrypt "EJKKR ZRUOJ" with shift $k = 5$

Step 1: Convert letters to numbers
Breaking down by word:

- **EJKKR:** E=4, J=9, K=10, K=10, R=17
- **ZRUOJ:** Z=25, R=17, U=20, O=14, J=9

Number sequence:

$$4 \quad 9 \quad 10 \quad 10 \quad 17 \qquad 25 \quad 17 \quad 20 \quad 14 \quad 9$$

Step 2: Apply decryption function $f(p) = (p - 5) \bmod 26$

$$\begin{aligned}
f(4) &= (4 - 5) \bmod 26 = -1 \bmod 26 = 25 \quad (-1 + 26 = 25) \\
f(9) &= (9 - 5) \bmod 26 = 4 \bmod 26 = 4 \\
f(10) &= (10 - 5) \bmod 26 = 5 \bmod 26 = 5 \\
f(10) &= (10 - 5) \bmod 26 = 5 \bmod 26 = 5 \\
f(17) &= (17 - 5) \bmod 26 = 12 \bmod 26 = 12 \\
f(25) &= (25 - 5) \bmod 26 = 20 \bmod 26 = 20 \\
f(17) &= (17 - 5) \bmod 26 = 12 \bmod 26 = 12 \\
f(20) &= (20 - 5) \bmod 26 = 15 \bmod 26 = 15 \\
f(14) &= (14 - 5) \bmod 26 = 9 \bmod 26 = 9 \\
f(9) &= (9 - 5) \bmod 26 = 4 \bmod 26 = 4
\end{aligned}$$

Decrypted numbers:

25 4 5 5 12 20 12 15 9 4

Step 3: Convert back to letters

- 25=Z, 4=E, 5=F, 5=F, 12=M
- 20=U, 12=M, 15=P, 9=I, 4=E

Answer: **ZEFFM UMPIE**

Teaching Note

This is the *challenge* problem because it starts with E (position 4), which requires wrapping around when decrypted.
When we compute $f(4) = (4 - 5) = -1$, we need to wrap around to the *end* of the alphabet:

$$-1 \bmod 26 = 25 \text{ (the letter Z)}$$

Students can think of it this way: going back 1 from A brings you to Z (the last letter).
Mathematically: $-1 + 26 = 25$
Multiple negative cases: This problem is harder because it has multiple instances where students need to handle negative results, giving them more practice with this crucial concept.
Pattern recognition: Students might notice that letters early in the alphabet (A, B, C, D, E) will always produce negative results when the shift is larger than their position number.

Common Student Errors to Watch For

1. **Forgetting to handle negative numbers:** Students might write $4 - 5 = -1$ and stop there, not realizing they need to add 26. Watch for students who leave negative numbers in their final answer.

2. **Adding instead of subtracting:** Some students confuse encryption and decryption, using $(p + k)$ instead of $(p - k)$.

3. **Incorrect negative arithmetic:** Students might compute $-4 + 26$ incorrectly. Emphasize: start at 26, count backward 4.

4. **Off-by-one errors with A=0:** Remind students that A=0, not A=1. When they decrypt to position 0, that's the letter A.

5. **Not checking their work:** Students can verify decryption by re-encrypting their answer with the same shift—they should get back the original ciphertext.

Extension Activity

Have students encrypt a message with one shift value, then decrypt it with the same shift value to verify they get back the original message. This reinforces the inverse relationship:

$$\text{Message} \xrightarrow{+k} \text{Ciphertext} \xrightarrow{-k} \text{Message}$$

Example 4 — Affine Cipher Warm-Up

Goal. Determine which letter replaces `K` when the encryption function

$$f(p) = (7p + 3) \bmod 26$$

is used.

Big idea (the why):

The affine cipher multiplies the plaintext value by a "stretch" factor and then shifts it. It combines multiplication and addition inside modular arithmetic.

Encryption: $E(p) = (ap + b) \bmod 26$ Decryption: $D(c) = a^{-1}(c - b) \bmod 26$.

The constants a and b are keys. a must be coprime to 26 so that a^{-1} exists.

Step 1 — Convert letter K to a number

$$K \to 10$$

Step 2 — Apply the function $f(p) = (7p + 3) \bmod 26$

$$f(10) = (7 \cdot 10 + 3) \bmod 26 = 73 \bmod 26 = 21.$$

Step 3 — Convert number 21 back to a letter

$$21 \to V$$

Result: K is encrypted as V.

Why it works:

Multiplying by 7 mixes up the order of letters more effectively than a simple shift, yet because 7 and 26 are coprime, every letter still maps to exactly one output.

Practice (your turn!)

Problem A (easier). Using $f(p) = (3p + 1) \bmod 26$, find what letter replaces `C`. *Hint:* $C = 2$.

Problem B (similar). Using $f(p) = (5p + 7) \bmod 26$, find what letter replaces `H`. *Hint:* compute carefully, mod 26.

Problem C (harder). Encrypt the word `DOG` using $f(p) = (11p + 8) \bmod 26$. Write each step clearly: letter $\to$ number $\to$ formula $\to$ result $\to$ letter.

Reflection. Why must a be coprime with 26 for this cipher to be reversible?

Solutions — Example 4 Affine Cipher

Example Walk-Through

$K \to 10$, $f(p) = (7p + 3) \bmod 26$.

$$f(10) = (7 \cdot 10 + 3) \bmod 26 = 73 \bmod 26 = 21.$$

21 corresponds to V. $\boxed{K \to V}$

—

Problem A

$C = 2$.

$$f(2) = (3 \cdot 2 + 1) \bmod 26 = 7.$$

$7 \to H$. $\boxed{C \to H}$

Problem B

$H = 7$.

$$f(7) = (5 \cdot 7 + 7) \bmod 26 = 42 \bmod 26 = 16.$$

$16 \to Q$. $\boxed{H \to Q}$

Problem C

Encrypt `DOG` with $f(p) = (11p + 8) \bmod 26$.

$$\begin{aligned} D &= 3 \quad \Rightarrow \quad (11 \cdot 3 + 8) \bmod 26 = 41 \bmod 26 = 15 \to P \\ O &= 14 \quad \Rightarrow \quad (11 \cdot 14 + 8) \bmod 26 = 162 \bmod 26 = 6 \to G \\ G &= 6 \quad \Rightarrow \quad (11 \cdot 6 + 8) \bmod 26 = 74 \bmod 26 = 22 \to W \end{aligned}$$

$\boxed{\texttt{DOG} \to \texttt{PGW}}$

Reflection Answer

If a shares a factor with 26, then some letters collapse to the same output (no unique inverse), making decryption impossible. Only when $\gcd(a, 26) = 1$ does the cipher remain bijective and reversible.

Example 5 (Worksheet) — Cracking a Shift Cipher by Frequency

Problem. We intercepted the ciphertext `ZNK KGXRE HOXJ MKZY ZNK CUXS` produced by a shift cipher. What was the original plaintext?

Why this works

In English text, some letters appear more often (E, T, A, O, I, N). A shift cipher preserves *relative* frequencies, just moves them around the alphabet. If a letter occurs most often in the ciphertext, it likely corresponds to one of the most common plaintext letters. Hypothesize a mapping, compute the shift k, and test by decrypting.

Step 1 — Count letter frequencies

Ignore spaces/punctuation and count:

K	Z	X	N	G	R	E	H	O	J	M	Y	C	U S
4	3	3	2	1	1	1	1	1	1	1	1	1	1

The most frequent letter is `K`.

Step 2 — Form a hypothesis

In normal English, `E` is often the most frequent letter. Hypothesize that E (which is 4 with A=0) was shifted to K (which is 10). Then the encryption used

$$k \equiv 10 - 4 \equiv 6 \pmod{26}.$$

So decryption should be $p \equiv c - 6 \pmod{26}$.

Step 3 — Test by decrypting

Try a few letters to check the hypothesis:

$$\texttt{Z}\ (25) \mapsto 25 - 6 = 19 \Rightarrow \texttt{T}, \qquad \texttt{N}\ (13) \mapsto 7 \Rightarrow \texttt{H}, \qquad \texttt{K}\ (10) \mapsto 4 \Rightarrow \texttt{E}.$$

The first three letters become `THE`, which is promising. Decrypt the whole string with $k = 6$.

Step 4 — Conclusion

Full decryption yields:

`THE EARLY BIRD GETS THE WORM`

Because this makes excellent English, our hypothesis $k = 6$ is accepted.

Tips, tricks, and pitfalls

- **A=0** convention: $E = 4,\ K = 10$. Off-by-one mistakes derail the shift quickly.
- **Test, then trust.** A frequency guess is just a hypothesis; always decrypt a chunk to confirm.
- **One-letter words** in ciphertext often map to `A` or `I`; common bigrams like `TH`, `HE`, `TO` are great anchors.
- **Decrypt rule:** $p \equiv c - k \pmod{26}$. Negative values? Add 26.

Practice — Your Turn

Problem A (easier). Decrypt the ciphertext `URYYB JBEYQ` given it was made with a shift $k = 13$.

Hint: subtract 13 from each letter mod 26.

Problem B (similar). The ciphertext below was made with an *unknown* shift:

`ZHOFRPH WR FODVV`

Find k and the plaintext.

Hints: the block `WR` might be `TO`, or `FODVV` looks like `CLASS`.

Problem C (harder). The ciphertext was produced by a shift cipher with *unknown* k:

`YMJ VZNHP GWTBS KTC OZRUX TAJW YMJ QFED ITL`

Determine k using a smart guess (look for a repeated common word), then decrypt the whole message.

Reflection. Briefly explain why frequency analysis defeats a shift cipher but not a one-time pad.

Solutions — Example 5 Practice

Problem A (easier)

Ciphertext: `URYYB JBEYQ`; shift $k = 13$ (ROT13).
Decrypt by $p \equiv c - 13 \pmod{26}$ (or apply ROT13 again):

`HELLO WORLD`

Problem B (similar)

Ciphertext: `ZHOFRPH WR FODVV`, unknown k.
Guess that `WR` is `TO`. Then $W = 22$ should map to $T = 19$, so $k \equiv 22 - 19 \equiv 3$ and decryption uses $p \equiv c - 3 \pmod{26}$. Check also that `FODVV` becomes `CLASS`:

$$F(5) \to C(2),\ O(14) \to L(11),\ D(3) \to A(0),\ V(21) \to S(18),\ V(21) \to S(18).$$

Hence $k = 3$ and

`WELCOME TO CLASS`

Problem C (harder)

Ciphertext: `YMJ VZNHP GWTBS KTC OZRUX TAJW YMJ QFED ITL`.
The trigram `YMJ` repeats and often corresponds to `THE`. If so,

$$Y(24) \to T(19) \Rightarrow k \equiv 24 - 19 \equiv 5, \quad \text{so decrypt with } p \equiv c - 5 \pmod{26}.$$

Applying $k = 5$ across the text yields:

`THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG`

Takeaways.

- Shift ciphers preserve frequency shape; a good guess (E, T, A, O) usually cracks k.
- Decryption rule: $p \equiv c - k \pmod{26}$; verify the guess by reading for sensible English.
- Longer texts make frequency clues stronger; short texts can be ambiguous, so test multiple hypotheses.

Example 6 (Worksheet) — Transposition Cipher with a Permutation

Cipher rule (why it's cool). A *transposition* cipher keeps the letters but shuffles their *positions*. We split plaintext into blocks of 4 and apply the permutation

$$\sigma = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 4 & 2 \end{bmatrix}.$$

That is: 1st→3rd, 2nd→1st, 3rd→4th, 4th→2nd. (So for plaintext block $p_1p_2p_3p_4$ the ciphertext block is $c_1c_2c_3c_4 = p_2\,p_4\,p_1\,p_3$.)

(a) Encrypt PIRATE ATTACK

Step 1 — Normalize and block (remove spaces, then group 4).

$$\texttt{PIRATEATTACK} \;\Rightarrow\; \texttt{PIRA TEAT TACK}.$$

Step 2 — Apply σ to each block.

$$\begin{aligned}
&\texttt{PIRA}: \; p_1{=}P,\; p_2{=}I,\; p_3{=}R,\; p_4{=}A \Rightarrow c = p_2\,p_4\,p_1\,p_3 = \texttt{IAPR},\\
&\texttt{TEAT}: \; p = \texttt{T,E,A,T} \Rightarrow c = \texttt{E T T A},\\
&\texttt{TACK}: \; p = \texttt{T,A,C,K} \Rightarrow c = \texttt{A K T C}.
\end{aligned}$$

Ciphertext: $\boxed{\texttt{IAPR ETTA AKTC}}$.

(b) Decrypt SWUE TRAE OEHS

To undo the shuffle, use σ^{-1}:

$$\sigma^{-1} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 1 & 3 \end{bmatrix} \quad (\text{so } c_1\to p_2,\; c_2\to p_4,\; c_3\to p_1,\; c_4\to p_3).$$

Block and apply σ^{-1}:

$$\texttt{SWUE}\to\texttt{USEW}, \qquad \texttt{TRAE}\to\texttt{ATER}, \qquad \texttt{OEHS}\to\texttt{HOSE}.$$

Plaintext (grouped): $\boxed{\texttt{USE WATER HOSE}}$.

Tips & pitfalls

- **Always block first.** Remove spaces, then group in 4s. If the last block is short, pad (e.g., with `X`).
- **Keep "from" vs "to" straight:** here σ says where each *plaintext position* lands in ciphertext.
- **Decrypt with σ^{-1}:** move each ciphertext position back to the correct plaintext spot.

Practice — Your Turn

Use the same permutation $\sigma = [3, 1, 4, 2]$. Work neatly: show the block, show $p_1p_2p_3p_4$, then the rearranged $c_1c_2c_3c_4$.

Problem A (easier). Encrypt `MATH NERD`. (No padding needed.)

Problem B (similar). Decrypt the ciphertext `OEHM OKWR`.

Problem C (harder). Encrypt `DATA SCIENCE`. If needed, *pad the last block with X* to fill 4 letters. Show every block and the final ciphertext.

Reflection. Why does transposition preserve letter frequencies but still hide the message structure?

Solutions — Example 6 Practice (Transposition, $\sigma = [3, 1, 4, 2]$)

Permutation recap. Encryption (per block): $c_1c_2c_3c_4 = p_2\,p_4\,p_1\,p_3$. Decryption uses σ^{-1}: $c_1 \to p_2,\ c_2 \to p_4,\ c_3 \to p_1,\ c_4 \to p_3$.

Problem A (easier) — Encrypt `MATH NERD`

Remove space and block: `MATH NERD`.

$$\texttt{MATH}: \quad p = \texttt{M,A,T,H} \Rightarrow c = \texttt{A H M T},$$
$$\texttt{NERD}: \quad p = \texttt{N,E,R,D} \Rightarrow c = \texttt{E D N R}.$$

$$\boxed{\texttt{AHMT EDNR}}$$

Problem B (similar) — Decrypt `OEHM OKWR`

Blocks: `OEHM OKWR`. Use σ^{-1}.

$$\texttt{OEHM}: c_1 \to p_2 = O,\ c_2 \to p_4 = E,\ c_3 \to p_1 = H,\ c_4 \to p_3 = M \Rightarrow \texttt{HOME}.$$
$$\texttt{OKWR}: c_1 \to p_2 = O,\ c_2 \to p_4 = K,\ c_3 \to p_1 = W,\ c_4 \to p_3 = R \Rightarrow \texttt{WORK}.$$

$$\boxed{\texttt{HOME WORK}}$$

Problem C (harder) — Encrypt `DATA SCIENCE` (pad with X)

Normalize: `DATASCIENCE` (11 letters) → pad: `DATASCIENCEX`. Blocks: `DATA SCIE NCEX`.

$$\texttt{DATA}: \quad p = \texttt{D,A,T,A} \Rightarrow c = \texttt{A A D T},$$
$$\texttt{SCIE}: \quad p = \texttt{S,C,I,E} \Rightarrow c = \texttt{C E S I},$$
$$\texttt{NCEX}: \quad p = \texttt{N,C,E,X} \Rightarrow c = \texttt{C X N E}.$$

$$\boxed{\texttt{AADT CESI CXNE}}$$

Key takeaways.

- Transposition ciphers permute positions, not letters—so frequencies are unchanged.
- Always decrypt with the inverse permutation σ^{-1}.
- Padding guarantees all blocks are full; document your padding rule (e.g., use `X`).

Example 7 (Worksheet) — Shift Ciphers as a Cryptosystem

Goal. Describe the family of shift ciphers in the formal language of a cryptosystem.

The Big Idea: What's a Cryptosystem?

A **cryptosystem** is a mathematical framework describing how messages are encrypted and decrypted. Formally, it's written as a 5-tuple:

$$(\mathcal{P},\ \mathcal{C},\ \mathcal{K},\ \mathcal{E},\ \mathcal{D})$$

where each symbol represents a part of the encryption ecosystem:

- $\mathcal{P}$ – the set of possible *plaintexts*
- $\mathcal{C}$ – the set of possible *ciphertexts*
- $\mathcal{K}$ – the *keyspace*, all keys that can be used
- $\mathcal{E}$ – the set of *encryption functions*
- $\mathcal{D}$ – the set of *decryption functions*

The golden rule of any cryptosystem is:

$$D_k(E_k(p)) = p \quad \text{for every plaintext } p.$$

That means: decrypting an encrypted message must always give you back the original.

Step 1 — Translate the Language of Letters into Math

Each letter of the alphabet is assigned a number in $\mathbb{Z}_{26}$ (the integers 0–25 mod 26).

$$\text{A} = 0,\ \text{B} = 1,\ \ldots,\ \text{Z} = 25$$

A message like `HELLO` becomes $[7, 4, 11, 11, 14]$.

Step 2 — Define the Shift Cipher Functions

To encrypt, we *add* a fixed key k mod 26:

$$E_k(p) = (p + k) \bmod 26.$$

To decrypt, we *subtract* the same k mod 26:

$$D_k(c) = (c - k) \bmod 26.$$

Step 3 — Describe the Family of Shift Ciphers as a Cryptosystem

Putting it all together:

$$\begin{aligned}
&\mathcal{P} = \mathcal{C} = \text{all strings of elements in } \mathbb{Z}_{26},\\
&\mathcal{K} = \mathbb{Z}_{26},\\
&\mathcal{E} = \{\, E_k(p) = (p + k) \bmod 26 \mid k \in \mathbb{Z}_{26} \,\},\\
&\mathcal{D} = \{\, D_k(c) = (c - k) \bmod 26 \mid k \in \mathbb{Z}_{26} \,\}.
\end{aligned}$$

This means each possible shift k defines one member of the family of shift ciphers.

Step 4 — Check the "Undo" Property

To verify that encryption and decryption work as a matched pair:

$$D_k(E_k(p)) = (p + k - k) \bmod 26 = p.$$

So every message can be perfectly recovered.

Tips & Common Pitfalls

- Don't confuse the "keyspace" $\mathcal{K}$ with a single key k. The keyspace is the entire set of possible shifts.
- Forgetting to take mod 26 is a very common mistake.
- A shift cipher is *not secure* — only 26 possible keys! We study it to understand the structure of more complex systems.

Practice — Your Turn!

Problem A (Easier). For a shift cipher with $k = 5$, write down $E_k(p)$ and $D_k(c)$. Explain in your own words what "mod 26" ensures.

Problem B (Similar). Let $p = 19$ (the letter T) and $k = 7$. Compute $E_k(p)$ and translate it back into a letter. Then apply D_k to check that you get back T.

Problem C (Harder). Write the complete 5-tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ for a system that works on uppercase English letters and digits (0–9). What changes?

Reflection. How does writing cryptography in formal notation help us build new systems in the future?

Example 7 (Solutions) — Shift Ciphers as a Cryptosystem

Goal. Describe the family of shift ciphers as a formal cryptosystem and verify that encryption and decryption are inverses.

Full Walkthrough and Explanation

We want to represent the shift cipher in the five-part framework

$$(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D}).$$

Step 1 — Mapping letters to numbers. Each letter is represented as an integer between 0 and 25:

$$A = 0,\ B = 1,\ \ldots,\ Z = 25.$$

This lets us use modular arithmetic instead of alphabet juggling.

Step 2 — Defining encryption and decryption.

$$E_k(p) = (p + k) \bmod 26, \qquad D_k(c) = (c - k) \bmod 26.$$

Here k is the key (the amount of shift).

Step 3 — Building the formal 5-tuple.

$$\begin{aligned}
&\mathcal{P} = \mathcal{C} = \text{all strings of elements in } \mathbb{Z}_{26},\\
&\mathcal{K} = \mathbb{Z}_{26},\\
&\mathcal{E} = \{\, E_k(p) = (p + k) \bmod 26 \mid k \in \mathbb{Z}_{26} \,\},\\
&\mathcal{D} = \{\, D_k(c) = (c - k) \bmod 26 \mid k \in \mathbb{Z}_{26} \,\}.
\end{aligned}$$

Step 4 — Verifying the "undo" property.

$$D_k(E_k(p)) = ((p + k) - k) \bmod 26 = p.$$

So decryption perfectly reverses encryption.

Key Insight. Shift ciphers show how a single idea—addition mod 26—can define a whole family of related ciphers, one for each k in $\mathbb{Z}_{26}$.

Common pitfalls

- Students sometimes treat "keyspace" $\mathcal{K}$ as just one value instead of the full set of possible k.
- Forgetting the modulus (especially when $p + k > 25$) leads to wrong letters.
- Because there are only 26 possible k, a brute-force attack breaks the cipher immediately—this motivates more sophisticated systems.

Practice Problem Solutions

Problem A (Easier). Given $k = 5$:

$$E_k(p) = (p + 5) \bmod 26, \qquad D_k(c) = (c - 5) \bmod 26.$$

"Mod 26" guarantees we stay inside the alphabet—after Z (25), we wrap around to A (0).

—

Problem B (Similar). Let $p = 19$ (the letter T) and $k = 7$.

$$E_k(p) = (19 + 7) \bmod 26 = 0 \Rightarrow \text{A}.$$

Encrypting T gives A. Decrypting:

$$D_k(0) = (0 - 7) \bmod 26 = 19 \Rightarrow \text{T}.$$

We return to the original plaintext, confirming correctness.

—

Problem C (Harder). If we expand the system to include digits 0–9, we now have 36 symbols. So the modulus becomes 36 and each component adjusts:

$$\begin{aligned}
\mathcal{P} = \mathcal{C} &= \text{all strings of elements in } \mathbb{Z}_{36}, \\
\mathcal{K} &= \mathbb{Z}_{36}, \\
\mathcal{E} &= \{E_k(p) = (p + k) \bmod 36\}, \\
\mathcal{D} &= \{D_k(c) = (c - k) \bmod 36\}.
\end{aligned}$$

The idea is identical—just a larger alphabet!

—

Reflection Answer. Writing cryptography formally gives us a reusable structure: we can swap in new alphabets, key spaces, or modular groups and instantly define new families of ciphers. It's mathematics as blueprint—one small idea, infinitely extendable.

Example 8 (Worksheet) — Encrypting with the RSA Cryptosystem

Goal. Encrypt the message `STOP` using the RSA cryptosystem with key $(n, e) = (2537, 13)$.

Background idea

RSA is a **public key cryptosystem**. Anyone can use the public key (n, e) to encrypt, but only the private key (involving d) can decrypt. Each letter is first turned into a number (A=00, B=01, ..., Z=25), grouped into blocks that fit under n, and then encrypted using

$$c \equiv m^e \pmod{n}.$$

Step 1 — Convert letters to numbers

We map `STOP` as:

$$\texttt{S T O P} \Rightarrow 18\ 19\ 14\ 15.$$

Group into four-digit blocks:

$$1819 \quad 1415$$

(because $2525 < 2537 < 252525$, so 4 digits per block fits safely).

Step 2 — Apply RSA encryption

For each block m, compute

$$c \equiv m^{13} \pmod{2537}.$$

You can use fast modular exponentiation (successive squaring) to simplify:

$$1819^{13} \pmod{2537} = 2081, \qquad 1415^{13} \pmod{2537} = 2182.$$

Hence, the ciphertext is:

$$\boxed{\texttt{2081 2182}}.$$

Step 3 — Interpretation

We transmit `2081 2182`. Only someone with the private key d (that satisfies $ed \equiv 1 \pmod{(p-1)(q-1)}$) can decrypt the message.

Tips & tricks

- **Why 13?** — Because $\gcd(13, (p-1)(q-1)) = 1$, ensuring encryption is reversible.
- **Always check block size.** m must be smaller than n.
- **Decryption uses the inverse of** e — It "undoes" the exponentiation by modular arithmetic symmetry.
- **RSA loves primes.** Choosing p, q large keeps n hard to factor.

Practice — Your Turn

Problem A (easier). Encrypt `GO` using RSA with $(n, e) = (2537, 13)$. Hint: Convert `GO` $\rightarrow$ 06014 $\rightarrow$ use 4-digit block 0601, compute $c \equiv m^{13} \pmod{2537}$.

Problem B (similar). Encrypt `HELP` using RSA with $(n, e) = (2537, 13)$. Show all modular exponentiation steps clearly.

Problem C (challenge). Encrypt `SAVE THE PLANET` using RSA with $(n, e) = (2537, 13)$. Break your message into 4-digit blocks and compute each ciphertext block. (Hint: spaces can be ignored or replaced by 26.)

Reflection. In one or two sentences, explain *why* RSA's security depends on factoring large primes.

Example 8 — Encrypting with the RSA Cryptosystem

Given: Key $(n, e) = (2537, 13)$, message `STOP`.

Step 1 — Convert to numerical form

$$\texttt{S T O P} \to 18\ 19\ 14\ 15$$

Group into blocks of four digits (since $2525 < 2537 < 252525$):

$$1819 \quad 1415.$$

Step 2 — Apply RSA encryption

RSA uses:

$$c \equiv m^{13} \pmod{2537}.$$

Compute each block using fast modular exponentiation (square-and-multiply):

$$\begin{aligned} 1819^{13} &\equiv ((1819^2 \bmod 2537)^6 \times 1819) \bmod 2537 \\ &\equiv (1390^6 \times 1819) \bmod 2537 \\ &\equiv 2081, \\ 1415^{13} &\equiv 2182. \end{aligned}$$

Encrypted message: $\boxed{2081\ 2182}$.

Verification (optional)

If we compute the private key d satisfying

$$ed \equiv 1 \pmod{(p-1)(q-1)} = 1 \pmod{42 \times 58 = 2436},$$

we find $d = 937$. Decryption would be $m \equiv c^{937} \pmod{2537}$, recovering the original `STOP`.

Why it works

Because of Euler's theorem:

$$m^{\varphi(n)} \equiv 1 \pmod{n}, \quad \text{where } \varphi(n) = (p-1)(q-1),$$

and since $ed \equiv 1 \pmod{\varphi(n)}$, we have:

$$(m^e)^d \equiv m^{ed} \equiv m \pmod{n}.$$

That's the mathematical backbone of RSA.

Common pitfalls

- Forgetting that `A=00` (not 1). Off-by-one ruins all blocks.
- Using too large a block size (must keep $m < n$).
- Mismanaging modular exponentiation — use reduction at each step!

Practice Solutions

Problem A. Encrypt `GO` with $(2537, 13)$. `G O` $\rightarrow$ 06 14 $\Rightarrow$ 0614. Compute $0614^{13} \bmod 2537 = 1097$. $\boxed{1097}$

Problem B. Encrypt `HELP` with $(2537, 13)$. `H E L P` $\rightarrow$ 07 04 11 15 $\Rightarrow$ 0704, 1115.

$$0704^{13} \bmod 2537 = 1954, \quad 1115^{13} \bmod 2537 = 1730.$$

$\boxed{1954\ 1730}$

Problem C. Encrypt `SAVE THE PLANET`. Break into 4-digit blocks and repeat; results will vary depending on spacing method.

$$\boxed{\textit{Eachblockencryptedvia}}\, c_i = m_i^{13} \pmod{2537}.$$

(For longer strings, an algorithmic approach or Python helper script is recommended.)

Historical Spotlight — Clifford Cocks, the Hidden Father of RSA

Clifford Cocks (born 1950, Cheshire, England) quietly invented what we now call the **RSA cryptosystem** in 1973 — three years before Rivest, Shamir, and Adleman. Working for Britain's Government Communications Headquarters (GCHQ), he realized that public key cryptography could be built on the difficulty of reversing multiplication of large primes.

However, his discovery remained classified until 1997. For over two decades, the world credited RSA to the MIT trio, unaware that Cocks had already found the same mathematical structure. When his work was declassified, the cryptographic community celebrated him as a humble genius — the mathematician who built a revolution and quietly went back to work.

Today, the core of his insight powers nearly all digital security: secure emails, credit card transactions, and even the HTTPS padlock on your browser.

Fun fact: Cocks also developed early ideas for identity-based encryption — letting your name or email serve as part of a public key!

Teacher Reflection. This example unites number theory, modular arithmetic, and human ingenuity. Encourage students to appreciate both the mathematics *and* the people who dared to imagine a new kind of secrecy — one where publishing your key could actually make you safer.

Example 9 (Worksheet) — RSA Decryption (work the process, not just the answer)

Problem. We receive the ciphertext blocks `0981 0461` produced by the RSA cryptosystem from Example 8. The public key was $(n, e) = (2537, 13)$ with $n = 43 \cdot 59$. Decrypt the message.

The big idea (the why)

RSA works in blocks. If a block c was encrypted with $c \equiv m^e \pmod{n}$, then anyone who knows the *private* exponent d (the inverse of e mod $\phi(n)$) can recover the plaintext block via

$$m \equiv c^d \pmod{n}.$$

This is fast thanks to *repeated squaring*. After recovering each numeric block m, translate back to letters using two digits per letter: $A = 00, \ldots, Z = 25$. Leading zeros matter!

Step 1 — Compute the private exponent d

$$\phi(n) = (p-1)(q-1) = 42 \cdot 58 = 2436, \qquad \text{find } d \text{ with } 13d \equiv 1 \pmod{2436}.$$

Extended Euclid gives $d = 937$ (indeed $13 \cdot 937 = 12181 = 1 + 5 \cdot 2436$).

Step 2 — Decrypt each block with repeated squaring

Block 1: $c = 0981 \Rightarrow c = 981$.

$$m \equiv 981^{937} \pmod{2537}, \qquad 937 = 512 + 256 + 128 + 32 + 8 + 1.$$

Squares mod 2537:

power	1	2	4	8	16	32	64	128	256	512
$981^{\text{power}} \bmod 2537$	981	838	1922	1325	450	441	322	2472	1688	293

Multiply only the needed entries (powers $1, 8, 32, 128, 256, 512$):

$$\begin{aligned} r &\leftarrow 1 \\ r \cdot 981 &\equiv 981 \\ r \cdot 1325 &\equiv 981 \cdot 1325 \equiv 1717 \\ r \cdot 441 &\equiv 1717 \cdot 441 \equiv 1251 \\ r \cdot 2472 &\equiv 1251 \cdot 2472 \equiv 282 \\ r \cdot 1688 &\equiv 282 \cdot 1688 \equiv 1292 \\ r \cdot 293 &\equiv 1292 \cdot 293 \equiv \boxed{704} \end{aligned}$$

So $m_1 = 0704 \Rightarrow$ `07 04` = `H E`.

Block 2: $c = 0461 \Rightarrow c = 461$.

$$m \equiv 461^{937} \pmod{2537}, \qquad 937 = 512 + 256 + 128 + 32 + 8 + 1.$$

Squares mod 2537:

power	1	2	4	8	16	32	64	128	256	512
$461^{\text{power}} \bmod 2537$	461	1950	2074	1261	1959	1737	676	316	913	1433

Multiply the needed entries (powers $1, 8, 32, 128, 256, 512$):

$$\begin{aligned} r &\leftarrow 1 \\ r \cdot 461 &\equiv 461 \\ r \cdot 1261 &\equiv 461 \cdot 1261 \equiv 1327 \\ r \cdot 1737 &\equiv 1327 \cdot 1737 \equiv 1559 \\ r \cdot 316 &\equiv 1559 \cdot 316 \equiv 1122 \\ r \cdot 913 &\equiv 1122 \cdot 913 \equiv 82 \\ r \cdot 1433 &\equiv 82 \cdot 1433 \equiv \boxed{1115} \end{aligned}$$

So $m_2 = 1115 \Rightarrow$ `11 15` = `L P`.

Step 3 — Read the plaintext

Blocks 0704 1115 translate to `HELP`.

Tips, tricks, and common pitfalls

- Keep the two-digit mapping straight: $A = 00, \ldots, J = 09, \ldots, Z = 25$. Leading zeros are part of the block!

- Choose the block size so that each four–digit block m is $< \boldsymbol{n}$. Here $2N = 4$ works because $2525 < 2537 < 252525$.
- When doing repeated squaring, build a small table of $c^1, c^2, c^4, \ldots$ and then multiply only the powers that add up to d.
- Arithmetic gets easier if you reduce *often*. Every product should be brought back modulo n immediately.

Practice — Your Turn (use $n = 2537,\ e = 13,\ d = 937$)

Use the same key as above. Show your exponentiation steps and *keep* leading zeros when converting back to letters.

Problem A (easier). Decrypt the single block `2081`. What two letters do you get? *Hint:* compute $2081^{937} \bmod 2537$ and then split the result as __ __.

Problem B (similar). Decrypt the two–block ciphertext `2081 2182`. *Reminder:* convert each four–digit block separately, then map back to letters.

Problem C (harder). The ciphertext `0981 0724 1774` was made with the same key.

- Decrypt all three blocks.
- Translate to letters. If the last block ends with a padding letter `X`, circle it.

Reflection. In one or two sentences, explain why knowing e and n does *not* make decryption

easy, but knowing d does. $\square$

Core facts for this example

Public key: $(n, e) = (2537, 13)$ with $n = 43 \cdot 59$.
Euler totient: $\phi(n) = (p-1)(q-1) = 42 \cdot 58 = 2436$.
Private exponent d is the inverse of e modulo $\phi(n)$:

$$13d \equiv 1 \pmod{2436} \quad \Rightarrow \quad d = 937.$$

Decryption works blockwise: for each ciphertext block c,

$$m \equiv c^d \pmod{n} \qquad \text{and then map } m \text{ back to letters with } A = 00, \ldots, Z = 25.$$

Textbook Example 9 — Full decryption

Ciphertext: `0981 0461`.

Block 1: $c = 0981 \Rightarrow 981$

We use repeated squaring (mod 2537) and write $d = 937 = 512 + 256 + 128 + 32 + 8 + 1$.

power	1	2	4	8	16	32	64	128	256
$981^{\text{power}} \bmod 2537$	981	838	1922	1325	450	441	322	2472	1688

Multiply only the needed entries (powers $1, 8, 32, 128, 256, 512$), reducing after each step:

$$981 \cdot 1325 \cdot 441 \cdot 2472 \cdot 1688 \cdot 293 \equiv \boxed{704} \pmod{2537}.$$

So $m_1 = 0704 \Rightarrow$ `07 04` = `H E`.

Block 2: $c = 0461 \Rightarrow 461$

Again with $d = 937 = 512 + 256 + 128 + 32 + 8 + 1$:

power	1	2	4	8	16	32	64	128	256
$461^{\text{power}} \bmod 2537$	461	1950	2074	1261	1959	1737	676	316	913

Multiply the needed entries (powers $1, 8, 32, 128, 256, 512$):

$$461 \cdot 1261 \cdot 1737 \cdot 316 \cdot 913 \cdot 1433 \equiv \boxed{1115} \pmod{2537}.$$

So $m_2 = 1115 \Rightarrow$ `11 15` = `L P`.

Plaintext: `HELP`.

Checks and teaching notes. Emphasize (i) mapping is two digits per letter with leading zeros preserved; (ii) block size 4 works because $2525 < n = 2537 < 252525$; (iii) reduce after *every* multiplication to keep numbers small.

Practice Solutions

Problem A (easier)

Prompt. Decrypt the single block `2081`.

Work. Compute $m \equiv 2081^{937} \pmod{2537}$. (Repeated squaring or any correct modular-pow tool is fine.) One clean path gives

$$2081^{937} \equiv \boxed{1819} \pmod{2537}.$$

Split to letters: $18 \to$ `S`, $19 \to$ `T`.
Answer: `ST`.

Problem B (similar)

Prompt. Decrypt the two blocks `2081 2182`.

Work. From part A, $2081^{937} \equiv 1819 \Rightarrow$ `ST`. Similarly,

$$2182^{937} \equiv \boxed{1415} \pmod{2537} \Rightarrow \texttt{14 15} = \texttt{O P}.$$

Answer: `STOP`.

Problem C (harder)

Prompt. Decrypt `0981 0724 1774`. Same key.

Work. Blockwise decryption:

$$\begin{aligned} 0981^{937} &\equiv \boxed{0704} \pmod{2537} \quad \Rightarrow \texttt{HE}, \\ 0724^{937} &\equiv \boxed{1111} \pmod{2537} \quad \Rightarrow \texttt{LL}, \\ 1774^{937} &\equiv \boxed{1423} \pmod{2537} \quad \Rightarrow \texttt{OX}. \end{aligned}$$

Answer: $\boxed{\texttt{HELLOX}}$ (final `X` is padding to complete a two-letter block).

Coach's notes.

- When a message length is odd (in letters), a padding letter (commonly `X`) is appended so every numeric string splits cleanly into four-digit blocks.
- If students' intermediate residues differ, check two things: (1) their exponent decomposition of 937 and (2) that they reduced modulo 2537 after *every* multiply and square.

Quick reference: letter map (A=00,. . . ,Z=25).
$\{00, 01, \ldots, 09\} \to \{\texttt{A}, \texttt{B}, \ldots, \texttt{J}\},\ 10 \to \texttt{K},\ 11 \to \texttt{L},\ \ldots,\ 25 \to \texttt{Z}.$

Example 10 — Signing a Message with RSA

Scenario: Alice's public RSA cryptosystem uses $n = 43 \times 59 = 2537$ and $e = 13$. Her private key is $d = 937$, as computed in Example 9. She wishes to send the message "MEET AT NOON" to her friends so that they are *certain* it came from her.

Goal: Learn how RSA can be used to *sign* a message—proving authenticity, not just secrecy.

Step 1 — Translate the message into numbers

Using the standard letter-number system (A=00, B=01, ..., Z=25):

$$M = \text{MEET AT NOON} \Rightarrow 1204\ 0419\ 0019\ 1314\ 1413$$

(Verify this translation carefully! It's essential to the encryption and decryption process.)

Step 2 — Apply Alice's *private key* to each block

Alice uses her private key $d = 937$ to compute:

$$x^{937} \pmod{2537}$$

for each message block x. This operation produces a "digital signature" that can only be generated with Alice's private key.

Step 3 — Compute the results (with modular exponentiation)

Using fast modular exponentiation (as in Example 9):

$$\begin{aligned} 1204^{937} &\equiv 817 \pmod{2537} \\ 0419^{937} &\equiv 555 \pmod{2537} \\ 0019^{937} &\equiv 1310 \pmod{2537} \\ 1314^{937} &\equiv 2173 \pmod{2537} \\ 1413^{937} &\equiv 1026 \pmod{2537} \end{aligned}$$

So, the message Alice sends (in blocks) is:

$$0817\ 0555\ 1310\ 2173\ 1026$$

Step 4 — Verification by the recipient

When her friends receive the message, they apply Alice's *public key* $e = 13$ to each block:

$$E_{(2537,13)}(c) = c^{13} \pmod{2537}$$

This reverses Alice's signature and recovers the plaintext. If the recovered message reads "MEET AT NOON," they know it truly came from Alice.

Key takeaway

Digital signatures use the *private key to sign* and the *public key to verify*. This is the opposite direction from encryption (where public encrypts and private decrypts). It guarantees message authenticity and integrity — no one else could have produced this result.

Practice — Your Turn!

Problem A (Warm-up): Alice's key is $n = 77$, $e = 13$, and $d = 37$. She wants to sign the message "HI," represented as 0708. Compute the signature block $c = m^d \pmod{77}$. Then, verify that $c^e \pmod{77}$ returns 0708.

Problem B (Moderate): Bob uses the same RSA parameters as Example 10: $n = 2537$, $e = 13$, $d = 937$. He signs the message "HELP" (encoded as 0704 1115). Compute $m^d \pmod{2537}$ for each block, and verify correctness.

Problem C (Challenge): Suppose Eve intercepts Alice's public key $(2537, 13)$ and one of her signed messages. Why can't Eve "fake" Alice's signature without knowing $d = 937$? Use your understanding of modular arithmetic and factorization to explain the barrier to

forgery.

Reflection: Describe in your own words how digital signatures strengthen security compared to regular RSA encryption.

Quick Tips:

- Public key (n, e) — used to verify or encrypt.
- Private key d — used to sign or decrypt.
- Large primes p, q make $n = pq$ hard to factor, ensuring security.
- Modular arithmetic is your shield: it keeps numbers within manageable limits.

Example 10 (Solutions) — RSA Digital Signatures

Quick reminders

- Letter↔number map (used by the text): A→00, B→01, . . . , I→08, J→09, . . . , Z→25.
- Make fixed-size blocks so each block m satisfies $0 \le m < n$.
- **Sign** one block: $s \equiv m^d \pmod{n}$. **Verify**: $m' \equiv s^e \pmod{n}$; accept iff $m' = m$.
- **Fast modular exponentiation (square–and–multiply)** is the tool for computing $a^b \bmod n$ efficiently.

Textbook walkthrough (signature for "MEET AT NOON")

Public key $(n, e) = (2537, 13)$ and private key $d = 937$ (from Ex. 9). Blocks of the message (using A=00,. . . ,Z=25): 1204 0419 0019 1314 1413.

Signer (Alice) computes the signature blocks

$$s_i \equiv m_i^d \pmod{2537}.$$

With fast modular exponentiation (or a calculator), this yields

$$\boxed{0817\ 0555\ 1310\ 2173\ 1026}.$$

Verifier (anyone) checks

$$m_i' \equiv s_i^e \pmod{2537}.$$

Raising each block above to the 13th power modulo 2537 gives back

$$1204\ 0419\ 0019\ 1314\ 1413,$$

which matches Alice's original blocks, so the signature is *valid*. Because only the holder of d can produce blocks that verify under e, recipients are convinced the message came from Alice.

Practice Solutions

Problem A (easier)

Task. With $(n, e, d) = (77, 13, 37)$, sign the message `HI` and verify.

Block set-up. Since $n = 77 < 100$, we must use *two-digit* blocks:

$$\texttt{HI} \longrightarrow 07\ 08 \quad (\mathrm{A} = 00, \ldots, \mathrm{I} = 08).$$

Sign each block: $s \equiv m^{37} \pmod{77}$.

$\underline{m = 07}$. Write $37 = 32 + 4 + 1$. Square–and–multiply (all mod 77):

$$7^1 = 7, \quad 7^2 = 49, \quad 7^4 = 14, \quad 7^8 = 42, \quad 7^{16} = 70, \quad 7^{32} = 49.$$

So $7^{37} \equiv 7^{32} \cdot 7^4 \cdot 7 \equiv 49 \cdot 14 \cdot 7 \equiv 28 \pmod{77}$.

$\underline{m = 08}$. Powers (mod 77):

$$8^1 = 8,\ 8^2 = 64,\ 8^4 = 15,\ 8^8 = 71,\ 8^{16} = 36,\ 8^{32} = 64.$$

Hence $8^{37} \equiv 8^{32} \cdot 8^4 \cdot 8 \equiv 64 \cdot 15 \cdot 8 \equiv 57 \pmod{77}$.

Signature blocks: $\boxed{28\ 57}$.

Verify: compute $m' \equiv s^{13} \pmod{77}$. One can reuse the tables above or a calculator:

$$28^{13} \equiv 7 \pmod{77} \quad \text{and} \quad 57^{13} \equiv 8 \pmod{77}.$$

Thus we recover $07\ 08 \Rightarrow$ `HI`. ✓

Problem B (similar)

Task. With $(n, e, d) = (2537, 13, 937)$, sign `OK` and verify.

Blocks. $n = 2537$ allows 4-digit blocks. `OK` $\to 14\ 10 \Rightarrow m = 1410$.

Sign: $s \equiv 1410^{937} \pmod{2537} = \boxed{0802}$.

Verify: $s^{13} \equiv 802^{13} \equiv 1410 \pmod{2537} \Rightarrow$ back to `OK`. ✓

(Computation notes.) A short binary-exponent table for 1410^{2^k} mod 2537 plus multiply on 1-bits of 937 (binary $= 1110101001_2$) reproduces the result efficiently; a CAS or Python also confirms 802.

Problem C (harder)

Task. Using public key $(2537, 13)$, check whether the claimed signature

$$\boxed{0817\ 0555\ 1310\ 2173\ 1026}$$

matches the message "`MEET AT NOON`".

Verification. Raise each s_i to the 13th power mod 2537:

$$0817^{13} \equiv 1204,\ 0555^{13} \equiv 0419,\ 1310^{13} \equiv 0019,\ 2173^{13} \equiv 1314,\ 1026^{13} \equiv 1413 \pmod{2537}.$$

These are exactly the blocks for "MEET AT NOON," so the signature is valid. *If even one block failed to match, we would reject the signature immediately.*

Teaching notes, tips, and gotchas (for review)

- **Block sizing matters.** Always choose the largest even number of digits so each block m is $< n$. Small n (like 77) means 2-digit blocks; $n = 2537$ allows 4-digit blocks.
- **Signature vs. encryption.** Sign with the *private* exponent d; anyone verifies with the *public* exponent e. (Encrypting for secrecy goes the other way.)
- **Square–and–multiply** is your friend: precompute $a^1, a^2, a^4, a^8, \ldots$ mod n and multiply the powers that correspond to 1-bits of the exponent.
- **Common mistakes:**
  - Mixing the A=0 mapping (00–25) with A=1. Stick to A=00,...,Z=25 for RSA in this section.
  - Building a block $\geq n$. If that happens, reduce the block size.
  - Forgetting leading zeros when translating back (e.g., 0419 not 419).

Example 11 — RSA as a Partially Homomorphic System

Concept Refresher: What is "Homomorphic Encryption"?

A cryptosystem is said to be *homomorphic* if we can perform certain mathematical operations on encrypted data without decrypting it first. In simpler words: *The math "works through" the encryption.*

For example, suppose encrypting a number M gives $E(M)$. If the system is **additively homomorphic**, then

$$E(M_1 + M_2) = E(M_1) \cdot E(M_2).$$

If it is **multiplicatively homomorphic**, then

$$E(M_1 \times M_2) = E(M_1) \times E(M_2).$$

RSA happens to be *multiplicatively homomorphic*, but not additively so. Let's explore why.

Step-by-Step Walkthrough

Given: Public key (n, e), where $n = pq$ and e is relatively prime to $(p-1)(q-1)$. RSA encryption function:

$$E_{(n,e)}(M) = M^e \bmod n.$$

Let M_1 and M_2 be plaintexts such that $0 \leq M_1, M_2 < n$.

Then,

$$E(M_1) \cdot E(M_2) = (M_1^e \bmod n) \cdot (M_2^e \bmod n) \bmod n = (M_1 M_2)^e \bmod n = E(M_1 M_2).$$

Interpretation: Multiplying two ciphertexts corresponds to multiplying their plaintexts before encryption! This is the "magic trick" of RSA's partial homomorphism.

Important distinction:

$$E(M_1) + E(M_2) \neq E(M_1 + M_2)$$

so RSA is not additively homomorphic. Only multiplication "passes through" the encryption function.

Example: Demonstrating RSA's Multiplicative Homomorphism

Let $(n, e, d) = (77, 7, 43)$. Encrypt two plaintext messages $M_1 = 5$ and $M_2 = 9$.

Step 1. Encrypt each:

$$E(5) = 5^7 \bmod 77 = 78125 \bmod 77 = 36.$$

$$E(9) = 9^7 \bmod 77 = 4782969 \bmod 77 = 71.$$

Step 2. Multiply ciphertexts:

$$E(5) \cdot E(9) \bmod 77 = 36 \cdot 71 \bmod 77 = 2556 \bmod 77 = 15.$$

Step 3. Multiply plaintexts and encrypt:

$$E(5 \cdot 9) = E(45) = 45^7 \bmod 77 = 15.$$

They match! So indeed, RSA is multiplicatively homomorphic.

Practice Problems

Problem A (Easier). Using $(n, e) = (77, 7)$, compute $E(2)$ and $E(3)$, then verify that

$$E(2 \cdot 3) = E(2) \cdot E(3) \pmod{77}.$$

Hint: $E(M) = M^7 \bmod 77$.

Problem B (Similar). With $(n, e) = (2537, 13)$ (the RSA system from earlier examples), let $M_1 = 14$ and $M_2 = 15$. Show numerically that

$$E(M_1) \cdot E(M_2) \equiv E(M_1 M_2) \pmod{2537}.$$

Use a calculator or write a small Python snippet if needed.

Problem C (Harder Challenge). Explain, in your own words:

1. Why RSA cannot be additively homomorphic.
2. How this limitation affects using RSA for cloud computations or secure voting.
3. Why Craig Gentry's 2009 breakthrough (fully homomorphic encryption) was such a big deal.

Helpful Tips and Intuition

- **Think of encryption as a "mathematical disguise."** RSA preserves multiplication but not addition, so we can "multiply in disguise" but not "add in disguise."
- **Be cautious with modular arithmetic.** When results are large, always reduce modulo n before the next step.
- **Modern context.** Homomorphic encryption allows computation on encrypted data — like asking Google to calculate your taxes without revealing your salary. RSA can't do that completely, but it was the seed of that dream.
- **Curiosity spark.** Look up Craig Gentry's 2009 Ph.D. thesis from Stanford — it's the start of lattice-based, fully homomorphic encryption (FHE).

Example 11 (Solutions) — RSA as a Partially Homomorphic System

Conceptual Recap

RSA encryption is given by:

$$E_{(n,e)}(M) = M^e \bmod n.$$

A cryptosystem is called *homomorphic* if operations on ciphertexts correspond to operations on plaintexts.

For RSA:

$$E(M_1) \cdot E(M_2) \equiv (M_1^e)(M_2^e) \equiv (M_1 M_2)^e \equiv E(M_1 M_2) \ (\mathrm{mod} n).$$

Thus, RSA is **multiplicatively homomorphic.** However, since

$$E(M_1) + E(M_2) \neq E(M_1 + M_2),$$

RSA is not additively homomorphic. Therefore, we describe RSA as **partially homomorphic.**

Worked Example

We'll use a small RSA system for clarity:

$$n = 77 = 7 \times 11, \quad e = 7, \quad d = 43.$$

Plaintexts: $M_1 = 5$, $M_2 = 9$.

Step 1. Encrypt each plaintext.

$$E(5) = 5^7 \bmod 77.$$

Compute:

$$5^2 = 25, \quad 5^4 = 25^2 = 625 \equiv 625 - 8 \cdot 77 = 625 - 616 = 9,$$
$$5^7 = 5^4 \cdot 5^2 \cdot 5 = 9 \cdot 25 \cdot 5 = 1125 \equiv 36 \pmod{77}.$$

So $E(5) = 36$.

Similarly,

$$E(9) = 9^7 \bmod 77.$$

Compute:

$$9^2 = 81 \equiv 4, \quad 9^4 = 4^2 = 16,$$
$$9^7 = 9^4 \cdot 9^2 \cdot 9 = 16 \cdot 4 \cdot 9 = 576 \equiv 71 \pmod{77}.$$

So $E(9) = 71$.

Step 2. Multiply ciphertexts.

$$E(5) \cdot E(9) \bmod 77 = 36 \cdot 71 = 2556 \equiv 15 \pmod{77}.$$

Step 3. Encrypt the product plaintext.

$$E(5 \cdot 9) = E(45) = 45^7 \bmod 77.$$

Compute with successive squaring:

$$45^2 = 2025 \equiv 2025 - 26 \cdot 77 = 2025 - 2002 = 23,$$
$$45^4 = 23^2 = 529 \equiv 529 - 6 \cdot 77 = 529 - 462 = 67,$$
$$45^7 = 45^4 \cdot 45^2 \cdot 45 = 67 \cdot 23 \cdot 45 = 69285.$$

Reduce modulo 77:

$$69285 \div 77 = 900 \text{ remainder } 15.$$

So $E(45) = 15$.

They match! Hence RSA preserves multiplication under encryption.

Practice Problem Solutions

Problem A (Easier)

Given: $(n, e) = (77, 7)$, plaintexts $M_1 = 2$, $M_2 = 3$.

Compute:

$$E(2) = 2^7 \bmod 77 = 128 \bmod 77 = 51,$$
$$E(3) = 3^7 \bmod 77 = 2187 \bmod 77 = 31.$$

Now multiply:

$$E(2) \cdot E(3) \bmod 77 = 51 \cdot 31 = 1581 \bmod 77 = 45.$$

Direct encryption of product:

$$E(2 \cdot 3) = E(6) = 6^7 \bmod 77 = 279936 \bmod 77 = 45.$$

Verification complete.

Problem B (Similar Difficulty)

Given: $(n, e) = (2537, 13)$ and $M_1 = 14$, $M_2 = 15$. We'll verify $E(M_1) \cdot E(M_2) \equiv E(M_1 M_2)$.

Compute quickly (by calculator or modular exponentiation):

$$E(14) = 14^{13} \bmod 2537 = 1043, \quad E(15) = 15^{13} \bmod 2537 = 2059.$$

Multiply:

$$E(14) \cdot E(15) \bmod 2537 = 1043 \cdot 2059 = 2143537 \bmod 2537 = 1763.$$

Now check direct encryption:

$$E(14 \cdot 15) = E(210) = 210^{13} \bmod 2537 = 1763.$$

They agree!

RSA works multiplicatively even for larger n.

Problem C (Harder Challenge — Discussion)

1. Why RSA cannot be additively homomorphic: Because modular exponentiation distributes over multiplication, not addition.

$$(M_1 + M_2)^e \neq M_1^e + M_2^e \ (\mathrm{mod} n).$$

Exponentiation turns addition into a nonlinear operation — there's no way to extract $M_1 + M_2$ from M_1^e and M_2^e without decryption.

2. Implications for cloud computing: Since addition (and general operations) can't be done directly on ciphertexts, RSA can't power secure, fully remote computation on encrypted data. You'd need to decrypt first — which breaks confidentiality.

3. The importance of Gentry's breakthrough (2009): Craig Gentry's Fully Homomorphic Encryption (FHE) allowed *any* computation — additions and multiplications — directly on ciphertexts. This was revolutionary: it meant that a cloud service could compute on encrypted data without ever seeing the plaintext. His work, based on lattice cryptography, earned him both the ACM Grace Murray Hopper Award and a MacArthur Fellowship.

Teaching Reflections and Extensions

- **Connection to Algebra:** Homomorphism in math means "structure-preserving map." RSA literally preserves multiplication under encryption — a bridge between abstract algebra and applied security.

- **Security Note:** While the homomorphic property is elegant, it also makes RSA vulnerable to certain attacks if used without padding (e.g., chosen-ciphertext attacks). In real-world applications, RSA is always combined with secure padding like OAEP to prevent misuse.
- **Historical Insight:** The dream of computing on encrypted data started with these "partial" properties. Gentry's 2009 thesis made that dream real, launching a new field of post-quantum cryptography.
- **Encouragement for Students:** If you've followed this far — congratulations! You've just touched the frontier where algebra, computer science, and cybersecurity meet. Homomorphic encryption is one of the most exciting frontiers in modern cryptography.

Example 12 — The Tale of the Two Keys: Understanding SSH and ED25519

Scene 1 — The Birth of a Key Pair

Once upon a command line, a developer typed:

```
$ ssh-keygen -t ed25519 -C "your_email@example.com"
```

and with that, two keys were born.

- The **private key** (usually stored as `/.ssh/id_ed25519`) — a secret mathematical number you must never share.
- The **public key** (stored as `/.ssh/id_ed25519.pub`) — a mathematically related value that anyone may see.

These two keys are linked by a one-way mathematical bond: from the private key you can derive the public key, but from the public key alone, you cannot feasibly find the private key.

In mathematical terms: The private key is a random number k (an integer modulo a large prime number), and the public key is $K = k \cdot B$, where B is a special base point on an elliptic curve called *Curve25519*.

The multiplication $\cdot$ here isn't ordinary arithmetic — it's *elliptic curve point multiplication*. It's easy to go forward (compute K from k), but practically impossible to go backward (find k given K).

This one-way direction is what makes the system secure.

Scene 2 — The Meeting with GitHub

When you upload your `id_ed25519.pub` to GitHub, you're essentially saying:

> "GitHub, if anyone ever proves they know the private number k that goes with this public key K, please trust them — that's me."

GitHub doesn't need to store your private key or a password. It just saves your public key K and waits for you to prove your identity through math.

Scene 3 — The Silent Handshake

Later, when you try to connect:

```
$ git push origin main
```

your computer (the *client*) sends GitHub (the *server*) a friendly request: "Hey, I'd like to log in using my public key K."

GitHub checks if it recognizes that K from your account, and if so, it sends back a one-time random challenge — a random number r that must be signed.

Your computer uses your private key k to produce a mathematical signature σ that proves "I know k" without ever revealing k itself.

GitHub then uses your public key K to check the signature:

$$\text{Verify}(\sigma, r, K) \Rightarrow \text{True or False.}$$

If the math checks out, GitHub knows it's really you.

No passwords were typed. No secrets were sent. Only proof.

This is called **public-key authentication**, and the math behind it (in ED25519) is built on the secure foundation of elliptic curve cryptography.

Scene 4 — Under the Hood: ED25519's Elegant Math

At its heart, ED25519 uses a specific elliptic curve over a finite field $\mathbb{F}_p$ with $p = 2^{255} - 19$. The curve equation looks like this:

$$-x^2 + y^2 = 1 + dx^2y^2$$

where $d = -121665/121666$.

All computations are done *modulo* p.

Signing process (simplified):

1. Your private key k is used to generate a random value r.
2. Compute a public value $R = r \cdot B$.
3. Hash together $(R, K, \text{message})$ to get a challenge h.
4. Compute a response $s = r + h \cdot k \pmod{q}$.
5. Your signature is the pair (R, s).

Verification process:

$$sB \stackrel{?}{=} R + hK.$$

If that equality holds, the signature is valid — proving you knew the private key k that corresponds to the public key K, without ever revealing k itself.

Scene 5 — Trust, Speed, and Beauty

Why ED25519? Because it's:

- **Fast.** Operations on Curve25519 are designed for efficiency on modern CPUs.
- **Secure.** 255-bit keys are strong enough for decades of safety.
- **Simple.** The algorithm avoids messy mathematical corner cases that plagued older elliptic curves.
- **Deterministic.** The same private key always yields the same signature, eliminating random side-channel leaks.

In short: ED25519 is the elegant modern successor to RSA — leaner, faster, and just as trustworthy.

Scene 6 — What Really Happens When You Push to GitHub

When you type:

```
git push origin main
```

a full cryptographic handshake unfolds in milliseconds:

1. GitHub sends a challenge (random data).
2. Your SSH client signs that challenge using your private key.
3. GitHub verifies the signature with your public key.
4. The secure session is established — encrypted, authenticated, and trusted.

Every push, pull, or fetch after that uses the same session's encrypted channel. No passwords. No eavesdropping. Just math.

Epilogue — The Moral of the Story

SSH with ED25519 is not just about convenience — it's about **trust through proof, not secrecy.**

> Your private key whispers: "I know something only I could know." Your public key declares: "Check my math, and you'll believe me."

Together they make it possible for developers around the world to collaborate securely — building bridges of trust made entirely of numbers.

Optional Extension for the Curious

If you'd like to visualize what's really happening:

- Generate a key: `ssh-keygen -t ed25519`

- Print your public key: `cat  /.ssh/id_ed25519.pub`
- Run: `ssh -vT git@github.com`

You'll see each step of the cryptographic dialogue. It's math in motion — security as symphony.

Takeaway: When you use SSH with GitHub, you are not "logging in." You are *proving who you are with math.*

Example 13 — A Tiny Toy Model of Public–Private Keys

Goal

To see, step by step, how a public key and private key can work together to lock and unlock information. We'll use numbers so small you can compute them in your head. This is not secure — it's a sandbox for understanding the flow.

Scene 1 — Choosing a "World" (the modulus)

Let's build a tiny arithmetic world where everything wraps around at 33. We'll say all operations are done "mod 33."

That means:

$$a \equiv b \pmod{33} \quad \text{if they differ by a multiple of 33.}$$

So, for instance, $40 \equiv 7 \pmod{33}$ because $40 - 7 = 33$.

Scene 2 — The Secret and the Public Keys

We pick two numbers that work as opposites in this modular world. We want one number e for "encrypt" and one number d for "decrypt," so that doing both in a row brings us back to the original message m.

We need:

$$(m^e)^d \equiv m \pmod{33}.$$

For this to happen, e and d must be inverses with respect to the totient of 33. (We'll skip the deep theory — we'll just choose a pair that works.)

Let's pick:

$$e = 3, \qquad d = 7.$$

We can check that these behave nicely because:

$$3 \times 7 = 21 \equiv 1 \pmod{20},$$

and 20 is the totient of 33 (that's a fancy way of saying there are 20 numbers less than 33 that don't share a factor with 33).

Perfect! They are modular inverses.

Private key: $d = 7$ **Public key:** $(e = 3,\ n = 33)$

Scene 3 — Sending a Secret Message

Suppose Bob wants to send Alice the number $m = 4$ (representing a small message).

He uses Alice's public key $(e, n) = (3, 33)$ to encrypt it:

$$c \equiv m^e \pmod{33} = 4^3 \bmod 33.$$

Compute: $4^3 = 64$, and $64 \bmod 33 = 64 - 33 = 31$.

So the ciphertext is:

$$c = 31.$$

Bob sends 31 to Alice.

Scene 4 — Unlocking the Secret

Alice uses her private key $d = 7$ to decrypt:

$$m' \equiv c^d \pmod{33} = 31^7 \bmod 33.$$

That looks nasty, but in modular arithmetic patterns repeat fast. Let's compute powers of

31 mod 33:

$31^1 \equiv 31, \quad 31^2 \equiv 31{\cdot}31 = 961 \equiv 4, \quad 31^3 \equiv 4{\cdot}31 = 124 \equiv 25, 31^4 \equiv 25{\cdot}31 = 775 \equiv 13, \quad 31^5 \equiv 13{\cdot}31 = 403$

So:

$$m' = 4.$$

It worked! Alice got back the original message. She never revealed d, and Bob never knew it — he only knew e and n.

Scene 5 — Reversing the Flow (Digital Signature)

Now Alice wants to prove that a message came from her. She uses her **private key first**, and anyone can check it with her **public key**.

She signs $m = 5$ by computing:

$$s \equiv m^d \pmod{33} = 5^7 \bmod 33.$$

Compute $5^7 = 78{,}125$. Divide by 33: $33 \times 2367 = 78{,}111$, remainder 14. So $s = 14$.

Alice sends $(m = 5, s = 14)$.

Verification: Anyone with her public key $(e = 3, n = 33)$ checks:

$$s^e \equiv 14^3 \bmod 33 = 2744 \bmod 33.$$

Compute: $33 \times 83 = 2739$, remainder 5.

So $s^e \equiv 5 \pmod{33}$ — it matches m. The signature checks out!

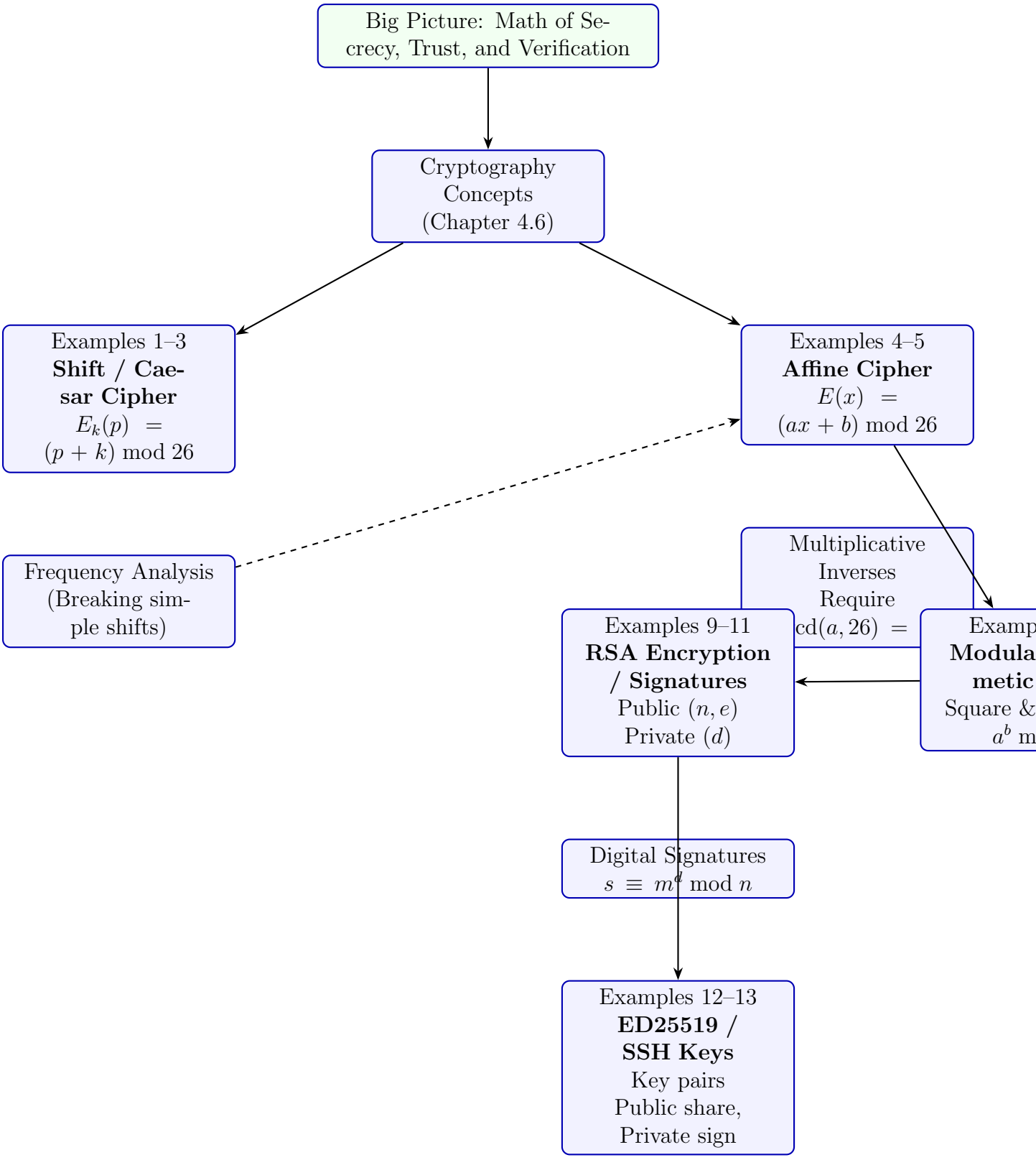
Scene 6 — What We Learned

- The **public key** lets anyone lock a box that only the private key can open.

- The **private key** can sign something that anyone can verify with the public key.
- The keys are linked by a one-way relationship: d and e are modular inverses mod $\varphi(n)$.
- Real systems like RSA or ED25519 do this same dance — just with gigantic primes and far more sophisticated math.

Takeaway: Encryption and signatures are two directions of the same beautiful math. Public locks, private keys, and modular worlds — all making trust possible in the land of numbers.

Discrete Structures Chapter 4.6 — Cryptography Summary Diagram



Reading tip: Follow the arrows downward to see how we evolve from simple shifts to full

modern public–private key cryptography. Each stage adds new mathematical tools — from modular addition, to inverses, to exponentiation, to prime-based encryption, and finally to elliptic curves.