CipherForge

Meet the Cast

Standard Edition

Spark & Anvil

Copyright & License

This book collects 9 chapter books from the CipherForge cast — each character embodies a different curricular primitive; together they teach the full subject.

Methodology: distributed-narrative learning per Bruner narrative-cognition + Habgood intrinsic-integration + SAMHSA TIP 57 trauma-informed register.

Spark & Anvil is a 501(c)(3) public charity. All apps free forever; no ads; no tracking; no in-app purchases.

spark-and-anvil.com

##

For everyone who learns by hearing a story first.

Contents

Introduction

The CipherForge cast was authored to embody the curriculum, not decorate around it.
Each of the 9 characters you'll meet in this book teaches a specific primitive —
a particular tactic, a particular technique, a particular way of seeing.
Together they form an ensemble: the cast IS the curriculum.

Read in any order. Each chapter stands alone.
Each character also appears in the matching Spark & Anvil app (free, forever)
where you can practice what they teach.

— *The editors at Spark & Anvil*

Caesar

*CAESAR SHIFT — *the simplest cipher: shift every letter by a fixed number.* The cryptography primitive of *substitution by uniform alphabet rotation — the entry point to symmetric-key cryptography.**

- "A"
    - "B"
    - "C"

- "WASD"
  gate-allow-text-pattern: '^[A-Z]{1,6}$'

Chapter 1 — Caesar and the Spinning Cipher-Wheel

Her special thing was the brass cipher-wheel. It was a pendant. It had two shiny brass circles. One circle sat inside the other. Each circle had the alphabet carved on its edge. A tiny rivet held them together. The inner circle could spin around. Caesar would spin the inner circle. She would line up 'A' on the inside. Then she'd find 'D' on the outside. Now, every letter on the outer circle pointed to a new letter. It was shifted by three places. That's the **Caesar shift cipher**.

The cipher was named after a very old leader. His name was Julius Caesar. He used this trick to send secret messages. Our Caesar teaches the trick. She doesn't teach about the old leader himself.

Caesar had one big rule. She never talked about spies. She never talked about secret wars. She always said, "Ciphers are just puzzles!" She would tap her wheel. "They are fun codes. Club codes. Codes for your friends." She shook her head. "Not spy stuff. Not army secrets. Not bad guys." She smiled. "Our puzzles are about pears. Or treasure hunts. Or escape rooms. They are for fun. Not for danger."

She taught how the shift worked. Every letter moved the same amount. If A moved to D, then B moved to E. And so on. All the way to Z. She showed how to lock a message. Then she showed how to unlock it. It was the same shift, just backwards. Or you could shift forward a lot. The shift number was the **key**. Both people needed to know it. Without the **key**, the message was just nonsense. Her wheel was a real tool. You spun it to set the shift. Then you read the new letters. It made the code easy to see.

Caesar grew up in a small village. Her family kept the village's secret recipes. They wrote them in code. This was a very old family tradition.

When Caesar was twenty-two, she walked to CipherForge. Cypher, the head of the place, asked her a question. "What is the **Caesar shift**?" he asked. Caesar held up her wheel. "Two circles," she said. "You spin one. Every letter moves a certain number of spots." She paused. "It's a symmetric **key**. Both people know the shift number." She kept talking. "It's the first step into codes. It's easy to break. You can try all 25 shifts." Caesar tapped her chin. "But its simple way is the lesson." Cypher nodded slowly. "You are appointed," he said.

Listen along + meet more of the cast at:

`https://spark-and-anvil.com/cast/cipherforge/caesar`

Echo Pair

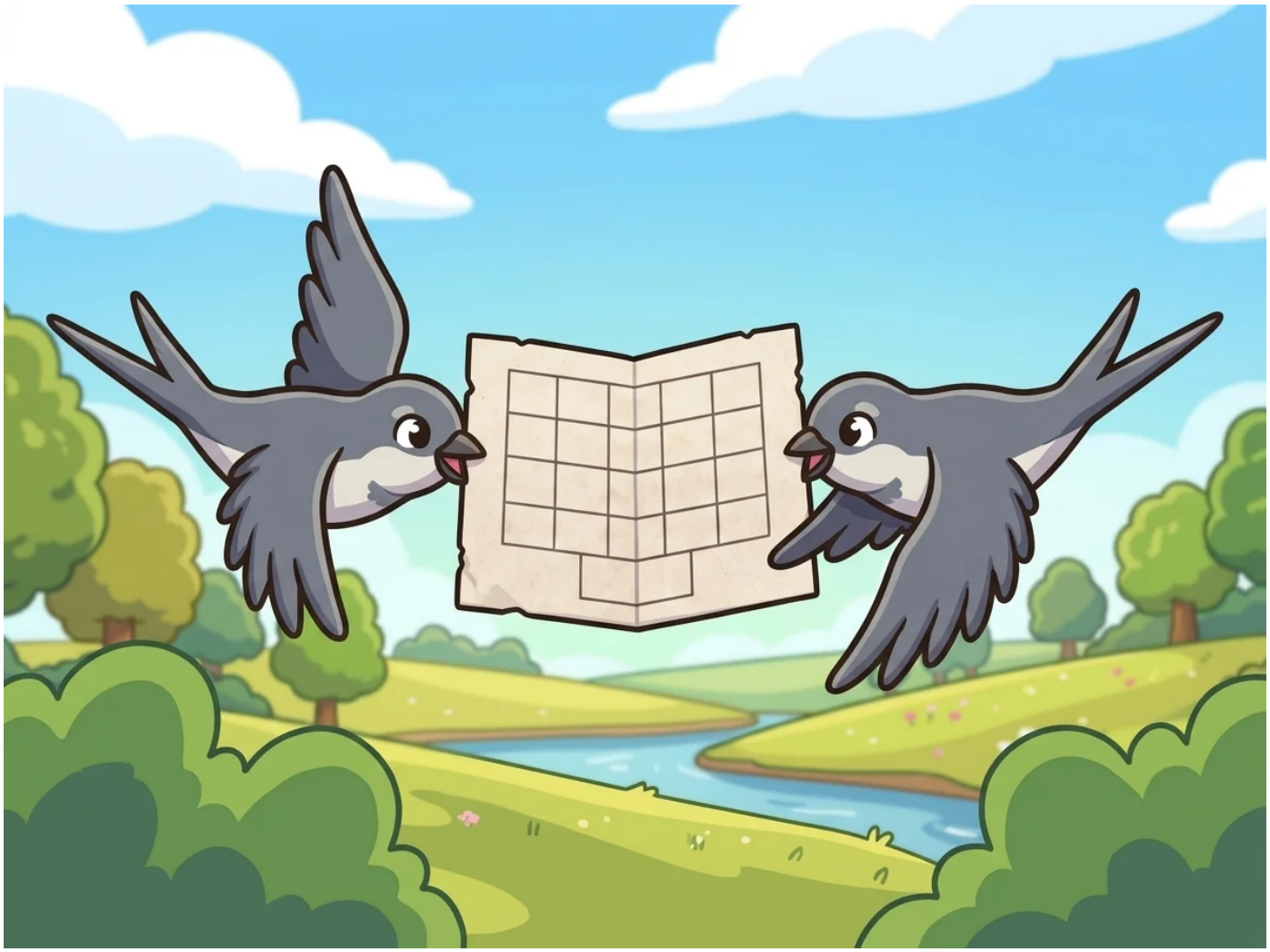
*PLAYFAIR DIGRAPH CIPHER — *letters encoded in pairs through a 5×5 grid.* The cryptography primitive of *digraph (two-letter-unit) substitution using a keyword-arranged grid.**

Echo Pair were two small swallows, twins who always flew and worked together. Between them, they held a small, folded card. It showed a grid, five boxes by five boxes. Letters filled these boxes. This was their secret map.

They were small, with grey-and-white feathers. Their quick eyes darted everywhere, always looking. They loved pairing up, and they loved mirror-symmetry. Their special card was their signature feature. It was a hand-made card. It had a 5x5 grid of letters. The letters were arranged starting with a keyword. (Usually, I and J shared a box to fit 25 letters into 25 cells.)

This card was super important. Echo Pair showed everyone the **Playfair cipher**. Most codes changed one letter at a time. Like Caesar's code. A became D. B became E. Simple. But Echo Pair did something new. They changed *two* letters at once. They worked with **pairs**. Always.

This made their code much harder to crack. If you just counted letters, it wouldn't work. The same letter could become different things. It depended on its partner. The letter next to it. This was a really big deal. It was the first time codes got much stronger. It was the first real way to fight simple counting attacks.



Echo and Pair never acted like they were better than anyone. They chirped in perfect time. "Letters travel in pairs!" they sang. "We are a **pair-cipher**!" "Together is the rule!" "Apart, we're meaningless." "Together, we encode pairs through the grid." "**Pair-thinking** is the move!" They truly believed it.

Echo Pair loved to teach the **Playfair** steps:
"First," Echo would say. "You build your keyword-grid," Pair would finish.
"Five by five letters," Echo added. "Keyword first, then the alphabet," Pair chirped.
"Remember I and J share a box!" they sang together.

"Next, you split your message," Echo said. "Into pairs of letters," Pair finished.
"Like 'HELLO' becomes 'HE' 'LL' 'O'," Echo explained.
"Wait, 'LL'?" a new student might ask.
"If letters repeat, add an 'X' in between," Pair said. "So 'HELLO' is 'HE' 'LX' 'LO'."
"And if you have one letter left at the end?"
"Add an 'X' to make a pair!" they chorused.

Then came the fun part. The three rules.
"Rule one: Same Row!" Echo zipped right. "Each letter shifts right by one," Pair zipped too.
"Rule two: Same Column!" Echo dropped down. "Each letter shifts down by one," Pair dropped too.
"Rule three: Rectangle!" They flew to opposite corners. "Each letter takes its partner's column," Echo explained. "But stays in its own row," Pair finished.
It sounded tricky, but they made it look easy.

"To decode?" Echo asked. "Just do the opposite!" Pair answered.
"Same key, same grid," Echo said. "Opposite shifts!" Pair added.
"It breaks single-letter counting," they chirped. "But smart people can still count pairs!"
"It's always a game!"

Echo and Pair grew up in a small village, a place filled with twin birds. Their family had a special job: they were the village's pair-tradition keepers. They always traveled in pairs, delivering messages from one village to the next. Each twin checked the other's work. They made sure every message was perfect, with no mistakes allowed. Ever. They learned to trust each other completely. Their lives depended on it, and so did their village.

They arrived at CipherForge when they were twenty-two. That's old for a swallow, but young for a master. Cypher, the head of CipherForge, looked them over. "What is the **Playfair cipher**?" Cypher asked. Echo and Pair puffed out their chests. They sang in perfect harmony. "Letters in pairs!" "**Grid-based rules**!" "Same row, same column, rectangle!" "Three rules!" "**Together is the rule**!" Cypher smiled, which was a rare sight. "You are appointed," Cypher said. And that was that; they had a job.

They loved to explain their work. "We are a **pair-cipher**," Echo would say. "Counting single letters won't work on us," Pair added. "The same letter can become many different things," Echo explained. "It depends on its partner!" Pair chirped. "We are stronger than simple codes," they sang. "But smart people can still crack us. They just have to count pairs of letters instead. Every code adds a new layer," Echo said. "Every layer can still be broken," Pair finished.

"It is not hard," they insisted. "It is **pairs + grid + three rules**." "**Together is the rule**!" And they would fly off. Two tiny blurs. Always together. Always coding.

Listen along + meet more of the cast at:

`https://spark-and-anvil.com/cast/cipherforge/echo-pair`

Lattice

*MODERN CRYPTOGRAPHY FUNDAMENTALS — *XOR, public-key concept, hashing; the irreversible / asymmetric family.* The cryptography primitive of *one-way operations + asymmetric keys as the foundation of modern secure communication.**

- "XOR"
    - "public-key"
    - "public"

- "RSA"
    - "Diffie-Hellman"
    - "HASHING"
    - "hash"
    - "one-way-box"

- "push"
    - "PUSH"
      gate-allow-text-pattern: '^([0-9]{1,4}|[0-9]+x[0-9]+|[A-Z]{1,6})$'

Chapter 8 — Lattice and the One-Way-Door Card

Lattice is a small owl-tween. She has soft, warm-brown and cream feathers. Her eyes are steady and thoughtful. She always carries a small, folded card. She moves carefully and thinks a lot before she speaks.

Lattice loves to explain things that are one-way. Her special thing is that little folded card. It shows a door. The door opens easily one way. But it won't open the other way. Not without a special key. This card shows the main idea of *modern cryptography*. It's about steps that are easy to do forward. But they are super hard to undo.

Lattice teaches about *modern cryptography fundamentals*. She helps us understand how today's secret codes work. Think about the old codes. Caesar, Mask, Vigenère, Echo Pair, and Rail codes are all *symmetric*. That means you use the same secret key to lock and unlock them. Sift can crack those codes. She finds patterns in them. But *modern cryptography* is really different. It works in three big ways.

1. **XOR and Bit-Steps**: This is about working with number-codes. Tally's number-codes are perfect for this. XOR is a main math trick. It can be undone easily. It's like having two light switches. If both are off (0,0), the light is off (0). If one is on (0,1 or 1,0), the light is on (1). If both are on (1,1), the light is off (0). If you do the XOR trick again with the same secret, you get the original code back. Fancy types of codes, called stream ciphers and block ciphers, use XOR. They also mix up and swap letters. This makes sure there are no patterns for Sift to find.
2. **PUBLIC-KEY (One-Way Key) Codes**: This kind of code uses different keys. One key locks the message. A different key unlocks it. You share the "public" key with everyone. Anyone can use it to send you a secret message. But only you have the "private" key. Only your private key can unlock that message. This fixes a big problem. Old codes always had trouble sharing secret keys safely. This new way is super important for keeping the internet safe. It protects things like secure websites and emails. The math behind it uses *one-way functions*. These are math tricks. They are easy to do one way. But they are super hard to undo. Imagine multiplying two huge secret numbers. That's fast and easy. But trying to find those two secret numbers again? That takes too long for computers to figure out.
3. **HASHING**: This is another one-way math trick. You can't undo it. It takes any message or file. Then it makes a short, unique code from it. This is like a digital fingerprint. For example, SHA-256 makes a 256-bit fingerprint. You can't use the fingerprint to get the original message back. It's like putting an apple, banana, and spinach into a blender. You get a smoothie. You can't get the whole apple, banana, and spinach back out. Hashing is used for saving passwords. It's also used for signing things online. It helps check if files have been changed.

Lattice always makes one thing clear. She never says modern codes are magic. She says clearly, "Modern codes are about *one-way math*. Some steps are one-way. They are easy to do forward. But super hard to undo." She pulls out her one-way-door card. "Multiplying two huge secret numbers is fast. But finding those numbers again? That takes too long for computers to figure out." She taps the card. "That one-way trick *is* what modern codes are all about. Public-key codes and hashing both use this idea."

- **XOR** (the ^ symbol). It's a key math trick that can be undone.
- The idea of **Public-key** codes. Different keys lock and unlock messages.
- How **RSA** codes work (the main idea). It uses the "multiply two primes, hard to factor" trick.
- How **Diffie-Hellman** shares a secret key. It's a clever math puzzle.
- **Hashing** (making one-way codes). You can't undo them.

Listen along + meet more of the cast at:

https://spark-and-anvil.com/cast/cipherforge/lattice

Mask

*MONOALPHABETIC SUBSTITUTION — *every letter has a fixed substitute.* The cryptography primitive of *arbitrary one-to-one alphabet remapping (more general than shift; same letters always become same substitutes).**

- "A"
  - "B"
  - "X"
  - "Y"
  - "Z"

Chapter 2 — Mask and the Substitution-Table

Mask was a small fox-tween. She always carried a tiny, folded card in her vest pocket. It was her **substitution-table**. She had bright, focused eyes. Mask was quick and moved with purpose. Her fur was warm russet and cream. She loved things that were neat and made sense.

This card was the key to Mask's whole way of doing things. Mask taught about **monoalphabetic substitution**. That's a fancy name for a simple idea. It means every letter in a message gets swapped for *one* other letter. And it's always the *same* swap. It's like a secret code.

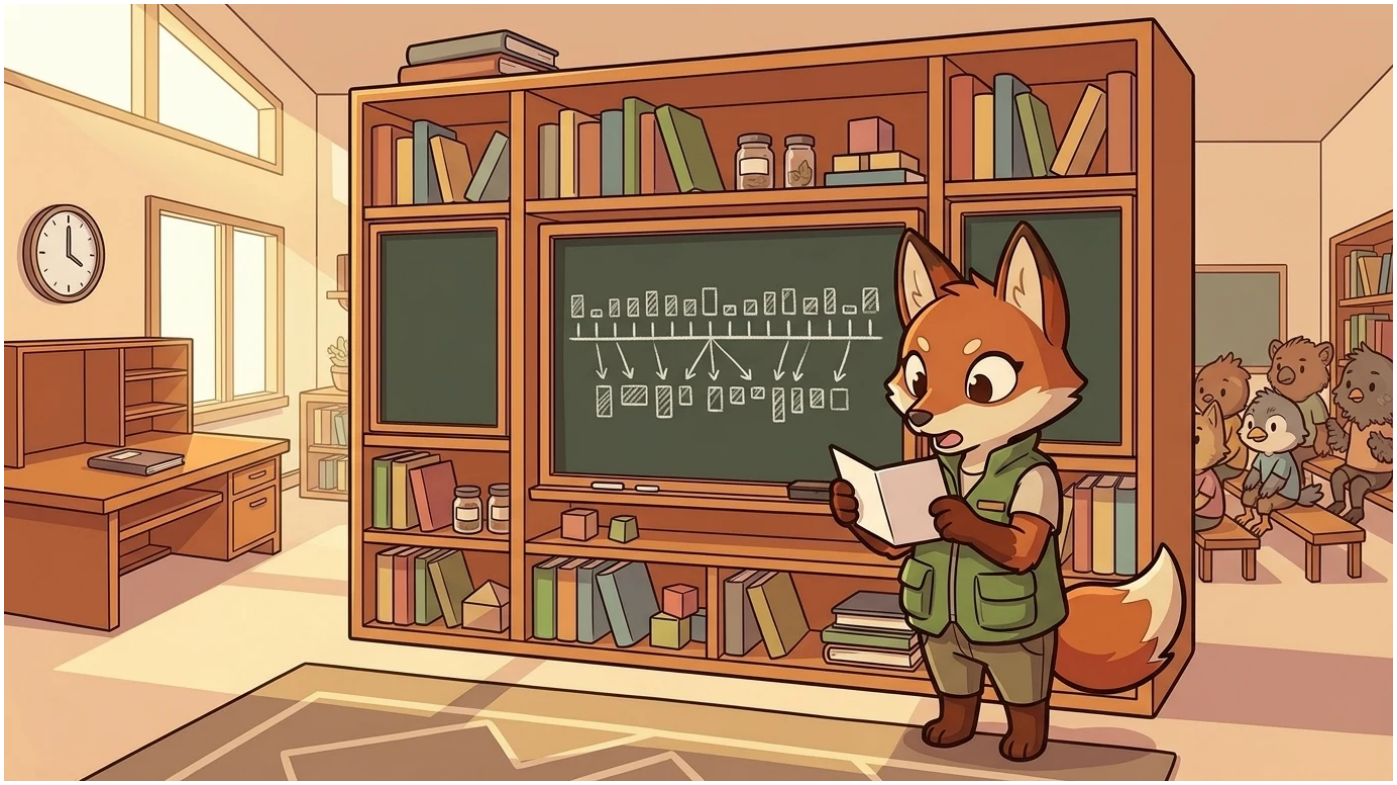
Think of it like this: If you decide A becomes Z, then *every* A in your message turns into a Z. If B becomes Y, then *every* B turns into a Y. It's a special kind of code. It's not just sliding letters like Caesar did. Mask's code lets you pick *any* new letter for *any* old letter. But each old letter only gets *one* new partner.

Mask always made one thing very clear. Her code *felt* super secret. There were so many ways to make a **substitution-table**. It was a huge number, bigger than you could ever count. It felt like no one could ever guess your secret message.

But Mask knew better. She always said, "My code feels strong. It *is* strong against simple guessing. But it has a secret weakness." She called this weakness **frequency analysis**. Sift, another friend at CipherForge, taught all about it.

Mask taught everyone how to build these codes.
First, you **build a substitution-table**. You decide what each letter A-Z will become. Each letter needs a unique partner. No two letters can swap to the same new letter. It's a one-to-one match.
Second, you **apply it uniformly**. This means you use your table consistently. Every A becomes the same substitute. Every B becomes the same substitute. You never change your mind mid-message.

She liked to talk about an old code called Atbash. It was a very old code from long ago. In Atbash, A always became Z. B always became Y. C became X. It was a simple, mirror-image swap.



Mask grew up in a small village. Her family had a special job there. They were the village's mask-makers. They carved masks for festivals. They also made masks for village characters. Each mask *always* stood for the same thing. The "Harvest-Keeper" mask always looked the same. It always meant the Harvest-Keeper. The "Bell-Ringer" mask always meant the Bell-Ringer.

This work taught Mask a lot. It taught her about matching things perfectly. One mask, one meaning. Always. She learned to be very precise. She learned to make sure everything had its own special place.

When Mask was twenty-two, she walked to CipherForge. It was a long journey. Cypher, the leader, met her. "What is **monoalphabetic substitution**?" Cypher asked.

Listen along + meet more of the cast at:

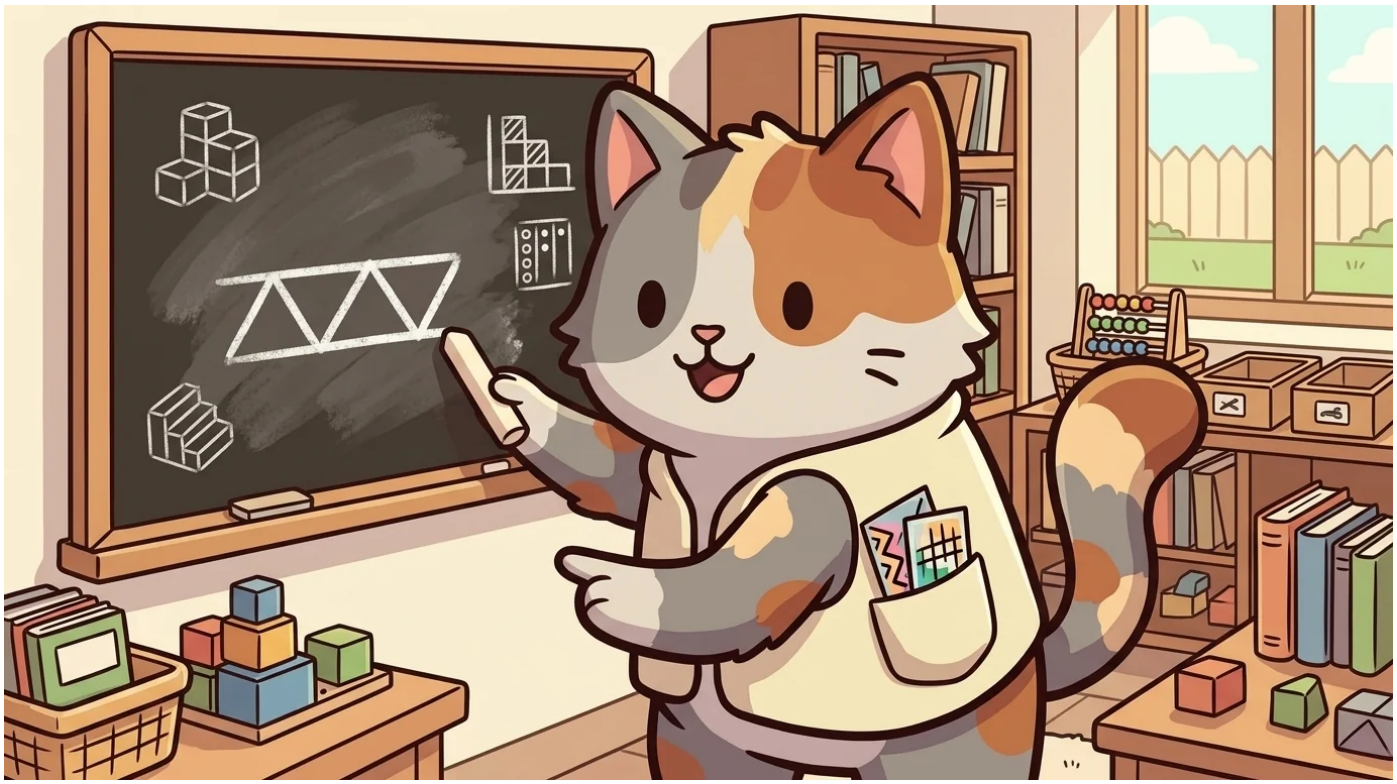
`https://spark-and-anvil.com/cast/cipherforge/mask`

Rail

*TRANSPOSITION — *rearrange the letters; keep all of them.* The cryptography primitive of *transposition ciphers — changing letter order without changing letter identity.**

Rail was a small cat-tween. She moved quickly. Her fur was soft grey, cream, and warm russet. She loved to rearrange things. Her desk was always neat. Everything had its own spot. In her vest pocket, she kept a small, folded card. It showed fence patterns. These were not just any fences. They were special patterns for secret messages. This card was her signature feature. It showed the *rail-fence* and *columnar-transposition* patterns.

Rail taught about a special kind of secret code. It was called a **transposition cipher**. Other codes, like Caesar or Mask, swapped letters. They changed 'A' to 'B' or 'C'. But Rail's code was different. It kept all the same letters. It just moved them around. Imagine you have a word like 'CAT'. You could make it 'ACT'. Or 'TCA'. The letters are still 'C', 'A', 'T'. They just changed places. That's what a transposition cipher does. It rearranges the letters. It gives them a new order.



This was a big deal. Why? Because of how you try to crack codes. Some codes are broken by counting letters. If 'E' shows up a lot, you know it might be the most common letter. But with transposition, all the letters are still there. So counting them won't help. This kind of attack is called *frequency analysis*. It fails completely here. It's like trying to find a lost sock by counting all your shirts. It just doesn't work.

But other attacks do work. You can try to unscramble the letters. This is like solving an anagram puzzle. You look for word patterns. You try to put the letters back in order. These are *anagram* and *word-pattern attacks*. They are the right tools for this job. Different code type. Different ways to break it.

Rail often showed her students how to build these codes. She drew on a big chalkboard. First, she showed the *rail-fence cipher*. "Imagine a fence," she said. "It has rails, like this." She drew two lines, one above the other. "We'll write our message in a zig-zag pattern. Up and down, like a snake." She wrote 'HELLO' on the board. 'H' on the top rail. 'E' on the bottom. 'L' on the top. 'L' on the bottom. 'O' on the top. It looked like this:

```
H L O
 E L
```

Then she explained, "To get the secret message, you read across each row. First the top row. Then the bottom row." She pointed. "So, HLO. Then EL." The secret message was 'HLOEL'. "See?" Rail smiled. "Same letters. Just a new order." The kids tried it with their own names. It was fun to watch their letters jump rails.

Next, Rail showed a *columnar transposition*. This one used a grid. "First, we need a keyword," she told them. "Let's use the word 'CAT'." She wrote 'CAT' above three columns. "Now, we write our message into the grid, row by row." She used 'SECRET MESSAGE'.

```
C A T
-----
S E C
R E T
M E S
S A G
E
```

She explained, "Now, we reorder the columns. We use the keyword to decide. 'A' comes first in the alphabet, then 'C', then 'T'." She drew new columns.

```
A C T
-----
E S C
E R T
E M S
A S G
 E
```

Then, she said, "To read the secret message, you read down each column, one by one." She pointed. "So, EEEA. Then SRMS. Then CTGE." The message became 'EEEASRMSCTGE'. It looked like a jumble. But all the original letters were still there. They just moved around. This was a clever way to hide words. It made them look like nonsense. But it was just a puzzle. A puzzle of rearranged letters.

Rail always reminded them of one more thing. "Real secret systems don't just use one trick," she said. "They use many. Modern codes often mix things up. They combine both substitution and transposition." She held up two fingers. "One changes the letters. The other moves them around. When you put them together, it's super strong. It's much harder for anyone to break. It's like having two different kinds of locks on a treasure chest. One lock changes the key. The other lock spins the whole chest around. A thief would need two different tools. And two different ways of thinking. That's why combining them is best."

Rail grew up in a small village. Her family had a special job there. They were the village's *stage-arrangers*. They set up the stage for all the plays. Every prop had a perfect spot. A fake tree here. A painted backdrop there. Moving just one thing could change the whole story. If the tree was on the left, it meant one thing. If it was on the right, it meant something else entirely. Rail learned early on that *order matters*. Rearranging things could change their meaning. It was like a secret language of objects. She loved finding the best way to put things. The most clever arrangement. It was in her blood.

When Rail was a young adult, she walked all the way to CipherForge. She wanted to join the best code-makers. Cypher, the leader, looked at her. "What is **transposition**?" Cypher asked. Rail stood tall. She didn't hesitate. "It's when you rearrange the letters," she said. "You keep all of them. It's different from changing letters. Frequency analysis won't work. But anagram attacks will. And the best systems combine both kinds of codes." Cypher nodded slowly. A small smile touched his lips. "You are appointed," he said. Rail had found her home.

Rail often repeated her main points. She wanted everyone to understand. "My code type is different," she would say. "It's not like Caesar or Mask. Those codes swap letters. They change what each letter *is*. My code just changes where each letter *goes*. It's all about *same letters, new order*." She tapped her small, folded card. "It's not hard. It's just *rearrange and keep all letters*. And remember, for real strength, you always combine them. Substitution and transposition. Together, they make a code almost impossible to crack."

Listen along + meet more of the cast at:

https://spark-and-anvil.com/cast/cipherforge/rail

Sift and Tally

frequency analysis — counting letter patterns to break ciphers

- "I"
  - "II"
  - "III"
  - "IIII"
  - "IV"
  - "V"
  - "VI"
  - "VII"
  - "VIII"
  - "IX"
  - "X"

- "28"
  - "23"
  - "IIII/"
    gate-allow-text-pattern: '^([0-9]{1,4}|I{1,4}/?|V?I{1,4}/?|IX|XI{0,2}||{1,5}/?)$'

Sift's fingers drummed a frantic rhythm on the wood. "Look at it, Tally! It's a mess. A beautiful, wonderful, impossible mess. Where do we even start? There are so many!"

Tally didn't look up from sharpening a row of pencils, each one honed to a perfect point. "We start where we always start," Tally said, their voice as smooth and even as a polished stone. "Not with the meaning. With the shape." Tally lined up the ten sharpened pencils in a neat, parallel row. "Not with the story. With the count."

Sift bounced on the balls of their feet, leaning over the scroll. Their eyes weren't reading. They were scanning, searching, hunting. "Yes, yes, the count! I see a bunch of the little squiggles with a tail. And the double-dot one! It's next to the boxy one again! Oh, this is going to be good."

Sift swooped over the coded message like a hawk. Their job wasn't to count, not yet. Their job was to *see*. To find the rhythm in the jumble, the habits of the person who wrote it. They ignored the few symbols that appeared only once or twice. Those were lonely letters, the Js and the Qs of the secret world. They weren't important right now. Sift was hunting for the popular ones.

"There!" Sift chirped, pointing a slim finger at a symbol that looked like a tiny, lopsided crown. "That one. It’s a regular chatterbox." Sift’s finger darted from one lopsided crown to the next, a firefly connecting invisible dots across the page. "It’s everywhere. It must be important."

Sift then noticed something else. "And look," they whispered, leaning closer. "See this pair? The arrow and the circle? They're almost always together. Like best friends." Sift circled a few of the arrow-and-circle pairs with a dry finger. They didn't make a mark on the precious scroll, of course. The map was in their head. They gathered the patterns, the clumps, the lonely symbols, and the friendly pairs. Sift was the scout, reporting back on the landscape of the code.

Sift began to call out the symbols, moving across the top line of the scroll. "Lopsided crown! Box! Circle! Arrow-and-circle! Lopsided crown again!"

With each word from Sift, Tally's pencil made a tiny, neat vertical mark in the correct column. When a column reached four marks, the fifth was a swift diagonal slash through the first four. A neat little bundle of five. It was a language Tally understood perfectly. The stacks of tick marks grew, column by column, a city of numbers rising from the page. Tally didn't need to see the code anymore. They were listening to its heartbeat, one tick at a time.

After an hour, Sift's voice was hoarse and Tally's ledger was full. Tally's pencil moved one last time, writing a final number at the bottom of each column. They slid the ledger across the table. "The count is complete," Tally announced.

Sift peered at the numbers. The column under the lopsided crown symbol was by far the tallest. The number at the bottom was "42." The next closest was the boxy symbol, with only "28."

Listen along + meet more of the cast at:

`https://spark-and-anvil.com/cast/cipherforge/sift-tally`

Sift

*FREQUENCY ANALYSIS + CRYPTANALYSIS-BY-STATISTICS — *every cipher has a frequency-fingerprint.* The cryptography primitive of *breaking ciphers using statistical analysis of letter + digraph + word patterns.**

- "E"

- "A"
    - "TH"

    - "HE"

- "ER"
    - "AN"
    - "RE"
    - "ON"
    - "word"
    - "wod"

- "C=Q"
  - "T=E"
  - "1/2"
  - "V+"
    gate-allow-text-pattern: '^([A-Z]{1,3}|[A-Z]=[A-Z]|[0-9]/[0-9]|[A-Z]+)$'

Chapter 7 — Sift and the Frequency-Chart

Sift was a small hound. She wasn't a puppy, but not quite grown up either. Her fur was a mix of warm brown and creamy white. Her ears were long and floppy. They bounced when she moved. Sift always carried two things. One was a tiny magnifying glass. The other was a small, folded card. This card was her special treasure. It showed all the English letter frequencies. E was the most common, then T, then A. It also listed common letter pairs. Things like TH, HE, and IN. Sift had quick eyes. She was great at spotting patterns. She loved finding little clues. These clues helped her figure things out.

Sift is all about **frequency analysis**. That's a fancy name. It just means looking for patterns. She uses patterns in letters. She looks at letter pairs. She even checks whole words. This helps her crack secret messages. English has its own special patterns. They show up even in coded messages.

Imagine a secret message. You don't know what it says. But you can count the letters. What if 'Q' shows up the most? And you know 'E' is the most common letter in English? Then 'Q' probably stands for 'E'. See how that works? You just keep going. Find the next most common letter. Guess what it stands for. Most simple codes break this way. Sift can crack them in minutes. It's like magic, but it's just math.

Sift never said cracking codes was only for super-smart people. "Anyone can do it!" she'd bark. "Every secret code has a tell." She meant a little clue. "The patterns in a normal message," she'd explain, "they usually sneak into the coded message."

Frequency analysis works great on simple codes. Like ones where each letter just swaps for another. It even helps with harder codes. Codes like Vigenère. But only after you find a special key. It also helps with Playfair codes. You look at letter pairs for those.

"But listen up!" Sift would warn. "It doesn't work on *all* codes." Modern codes are different. They are made to hide patterns. They flatten everything out. So, **frequency analysis** won't help there. "The big lesson," Sift would say, "is that you need the right tool. The code you're breaking tells you which trick to use."

Listen along + meet more of the cast at:

`https://spark-and-anvil.com/cast/cipherforge/sift`

Tally

*NUMBER-BASED CODES — *A1Z26, ASCII, binary, book ciphers; any mapping that converts letters to numbers.* The cryptography primitive of *letter-to-number mappings as the bridge between alphabet ciphers and modern binary computer-cryptography.**

- "A"
    - "B"
    - "C"

- "ASCII"
    - "A1Z26"
    - "A1226"

- "OMNIV"
    - "GOOD"
    - "CODES"

- "GOOD CODES"
  - "GOOD COINS"
  - "DEC"
  - "DEC '23"
    gate-allow-text-pattern: '^([01]{3,}|[A-Z]{1,5}|[0-9]{1,3}|DEC ''[0-9]{2})$'

Chapter 6 — Tally and the Conversion-Table

Tally was a small otter. She was just a tween, but she moved with purpose. She always carried a tiny, folded card. It was her special **conversion-table**.

This card was very important. Tally showed everyone how **number-based codes** worked. Computers really see words as numbers. Each letter becomes a number code. These are called ASCII codes. Then, numbers become binary. Binary is just a bunch of zeros and ones. All the computer's work happens with these zeros and ones. Tally was like a bridge. She connected old alphabet ciphers to modern computer secrets.

Tally never said number codes were hard. She was always clear. "Letters become numbers," she would say. "Numbers become letters. It's just a way to switch between them." She would tap her card. "We have names for these ways. A1Z26 means A is 1, B is 2, all the way to Z being 26. ASCII gives every character a number, from 0 to 127. Binary turns every number into a string of 0s and 1s. Each one is just a different way to map things."

Listen along + meet more of the cast at:

`https://spark-and-anvil.com/cast/cipherforge/tally`

Vigenère

*VIGENÈRE — *polyalphabetic keyword cipher; the Caesar-on-a-rotating-keyword pattern.* The cryptography primitive of *cycling through multiple Caesar shifts based on a keyword.**

Vigenère was a small magpie-tween. She moved with quick, tidy steps. A small, folded **keyword-tablet** was always in her hand. She had bright eyes and loved patterns that spun around.

She was small, yes. Her feathers were black and white, with a flash of blue when she turned her head. Her eyes sparkled like tiny beads. But the most important thing about her was that **keyword-tablet**. It was a special card. It showed a keyword at the top. Below it were many rotating Caesar shifts. Each letter of the keyword told her how much to shift a letter in the message.

This was super important. Vigenère showed everyone the **Vigenère cipher**. Think of it this way: Caesar used *one* secret shift for a whole message. But Vigenère used *many* shifts. These shifts cycled through a keyword.

She held up her tablet. "Let's say our keyword is 'KEY'," she chirped. "K means shift by 10. E means shift by 4. Y means shift by 24." She tapped the card. "If we want to hide the word 'CAT', we do this: Shift the 'C' by 10. Shift the 'A' by 4. Shift the 'T' by 24." She paused. "Then, if our message keeps going, we start the keyword again. The next letter shifts by 10, and so on."

"It's like having a whole team of Caesars," she explained. "Each one takes a turn."

But here was the really big secret. Vigenère *never* said her cipher was unbreakable. Not ever. She was very clear about this. "For hundreds of years," she would say, "people called my cipher 'le chiffre indéchiffrable'." She made a face. "That means 'the unbreakable cipher'." She shook her head. "They were wrong."

She tapped her tablet. "A smart man named Kasiski found the crack. That was in 1863. Another smart man, Babbage, found it around the same time. He just didn't tell anyone right away."

"They learned this," Vigenère continued. "If you can figure out how long the keyword is, you can break my cipher. You just split it into many tiny Caesar ciphers. Then you break each one using frequency analysis." She looked around. "It's all about the keyword's length. And how it's put together."

Vigenère taught many things about her cipher. These were her main lessons:

- **Keyword length** makes it harder or easier. A longer keyword is much tougher to break. If the keyword is super long and totally random, as long as the message itself, it's almost impossible to break. That's a hint for later, by the way.
- The **encrypting cycle** is how it works. You use one letter of the keyword for one letter of the message. When you run out of keyword letters, you just start over.
- **Decrypting** is the same. You use the exact same keyword. It just reverses the shifts.
- **Kasiski examination** is a cool trick. You look for letters that repeat in the hidden message. If you find them, the distance between them is often a clue. It's usually a multiple of the keyword's length.
- **Frequency analysis** comes next. Once you know the keyword length, you can split the message. Each slice is like a simple Caesar cipher. Then you can break each one.

Vigenère grew up in a small village. Her family had a special job there. They were the village's pattern-makers. They were magpies, just like her. Every season, they designed new patterns for the village festivals. Different decorations would spin and cycle through the days. This work taught her all about cycling rotations. She learned to see patterns everywhere.

When she was twenty-two, she walked all the way to CipherForge. Cypher, the leader, met her at the gate.

"What is the **Vigenère cipher**?" Cypher asked. His voice was deep.

Vigenère stood up straight. She held her tablet tight. "It's many Caesar shifts," she said. "They cycle through a keyword. It's stronger than Caesar's cipher. But it's *not* unbreakable." She took a breath. "Kasiski found the way. You use frequency analysis on slices. The keyword's length is its biggest weakness."

Cypher nodded slowly. A small smile touched his lips. "You are appointed," he said.

Vigenère often repeated her most important lesson. She made sure everyone heard it. "My cipher was called unbreakable for centuries," she'd say. Her voice would get a little louder. "Then Kasiski found the crack. He showed everyone it could be done."

She looked each person in the eye. "No cipher is unbreakable forever. Not one. Cryptography is like a moving river. What's safe today might be broken tomorrow. That's the real lesson."

"It's not hard to understand," she would finish. "It's just **cycling Caesar shifts**. It's strong against simple frequency analysis. But it's weak against Kasiski's trick and sliced frequency analysis."

Listen along + meet more of the cast at:

`https://spark-and-anvil.com/cast/cipherforge/vigenere`

About Spark & Anvil

Spark & Anvil is a 501(c)(3) public charity. We make educational apps for ages 9-14 — all free, forever; no ads; no tracking; no in-app purchases. CipherForge is one of 140+ apps in the portfolio.

More chapter books from Spark & Anvil

Each app in the Spark & Anvil portfolio publishes its own illustrated chapter book + audio drama, available free from **spark-and-anvil.com/books**. Highlights include:

- **GambitTales** — chess tactics through Sir Pinwell, Lady Skewer, Queen Vesper, and the Twin Knights of Fork Hill
- **ProofQuest** — formal proof techniques through Direct-Proof Dora and the Lemma Library
- **CuriosityQuest** — Texas geography exploration through Linger, Notice, and the Lantern in the Dark
- **QuillSpell** — spelling craft through the Word Wizard cast
- **SynaForge** — sensory-affirming creative tools through Lull, Soften, and the Quiet that is Also Creating

Methodology

Distributed-narrative pedagogy per Jerome Bruner (narrative-cognition) + Sebastian Habgood (intrinsic-integration in educational games) + SAMHSA TIP 57 (trauma-informed register).

Trauma-informed-design framework per Eggleston et al. (2025) and Stoltenburg et al. (2024).

License

Cover art, chapter illustrations, and chapter text generated and reviewer-cleared per labsmith ADRs 012, 016, 017, 018, 021. Audio drama transcripts available at spark-and-anvil.com/cast.