Kryptos K4: The ‘W’ as a Separator

Split at Its Five W’s, K4 Falls Into Two Groups With One Frequency Shape

Guillaume Lethuillier
glthr.com

March 30, 2024 · updated November 15, 2026

Series: Kryptos K4 — Part 1 of 2
Next: Kryptos K4: The W-Rectangle and Its Half-Turn (glthr.com/kryptos-k4-a-rectangle-hidden-in-the-cipher)
Interactive view: glthr.com/k4

Abstract

Split at its five W’s, Kryptos K4 falls into six segments which, dealt alternately into two groups, have the same letter-frequency shape. I measure how rare that is, show how such a cipher could be built, and present K4nundrum, the open-source tool I wrote to explore the idea. Note: for simplicity, in this article, I interchangeably use “encryption”, “encipherment”, and “encoding” to convey the idea of transforming the plaintext to the ciphertext (and vice versa, using their antonyms).

> **Update (November 2026):** This article records my observation of 2024. Its sequel, “Kryptos K4: The W-Rectangle and Its Half-Turn”, shows that the same five W’s, in rows of 19 letters, form a rectangle (interactive grid). It revises a claim made below: the W’s are most likely stops that stand for characters of the plaintext, not letters without a plaintext equivalent, so the plaintext would have 97 characters, not at most 92. In September 2025, journalists Jarett Kobek and Richard Byrne found the K4 plaintext among Jim Sanborn’s papers: it was located, not deciphered, and it has not been made public. The hypothesis presented below therefore remains untested.

K4, AS ON THE SCULPTURE

```
 1 OBKR 4
 5 UOXOGHULBSOLIFBBWFLRVQQPRNGKSSO 35
36 TWTQSJQSSEKZZWATJKLUDIAWINFBNYP 66
67 VTTMZFPKWGDKZXTJCDIGKUHUAUEKCAR 97
```

positions in the text

The K4 ciphertext: 97 letters.

```
OBKRUOXOGHULBSOLIFBBWFLRVQQPRNGKSSOTWTQSJQSSEKZZWATJKLUDIAWINFBNYPVTTMZFPK
↪ WGDKZXTJCDIGKUHUAUEKCAR
```

Cite as: Guillaume Lethuillier, “Kryptos K4: The ‘W’ as a Separator”, glthr.com, March 30, 2024. https://glthr.com/kryptos-k4-the-w-as-a-separator
Interactive figures, animations and videos of the web page are shown here as still images or addresses.

The "Kryptos" sculpture, dedicated in 1990 by its creator Jim Sanborn, contains four encoded messages. Cryptography enthusiasts have successfully decoded three of them. However, **the elusive fourth ciphertext, K4, remains undeciphered**, presenting a fascinating challenge to the global community of cryptanalysts.

Since 2010, Sanborn has offered clues (New York Times, Elonka Dunin's Kryptos page), hoping someone will solve it. To date, these cribs—as they are referred to within the field of cryptanalysis—are:

- The 22nd through 34th positions correspond to EASTNORTHEAST,
- The 64th through 74th positions correspond to BERLINCLOCK.

Despite these known plaintext words, K4 has yet to be deciphered. While my attempts have not yielded a solution, **I have just discovered a pattern that may provide valuable insights into the encryption method employed for K4**.[1]

I start with the bottom line. Then, I detail my approach and evaluate the relevance of this pattern.

The Signs of a Segmentation and Recombination Technique

I believe that:

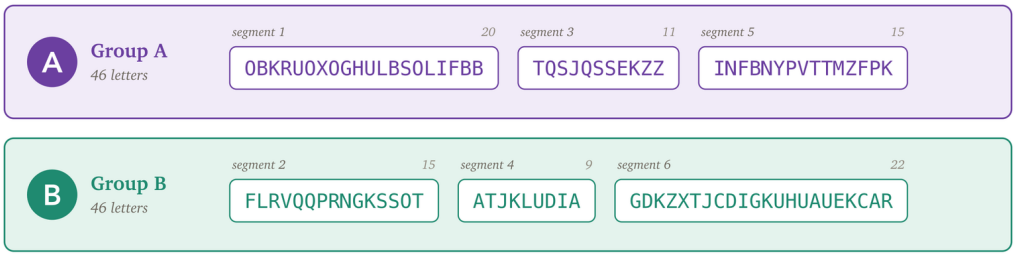
- **The letter 'W' acts as a separator that splits K4 into six segments**. (I define a *segment* as a contiguous series of characters that can be deciphered into a partial or a complete sentence, as in this fictitious example: HLMKZJZDDSABQ, a ciphertext segment, corresponding to BROWNFOXJUMPS, a plaintext segment.)



- Consequently, unlike other characters, the letter 'W' is not mapped to plaintext characters. Additionally, despite the ciphertext's total length of 97 characters, the plaintext can contain no more than 92 characters (97 − 5 occurrences of 'W').
- Each segment belongs to either of two groups (Group A and Group B).

(Update November 2026) Here I mean 92 letters of the message, excluding the five stops. Each W could stand for a stop, such as X for a full stop or ?. Including those five stop markers gives 97 plaintext characters in total.

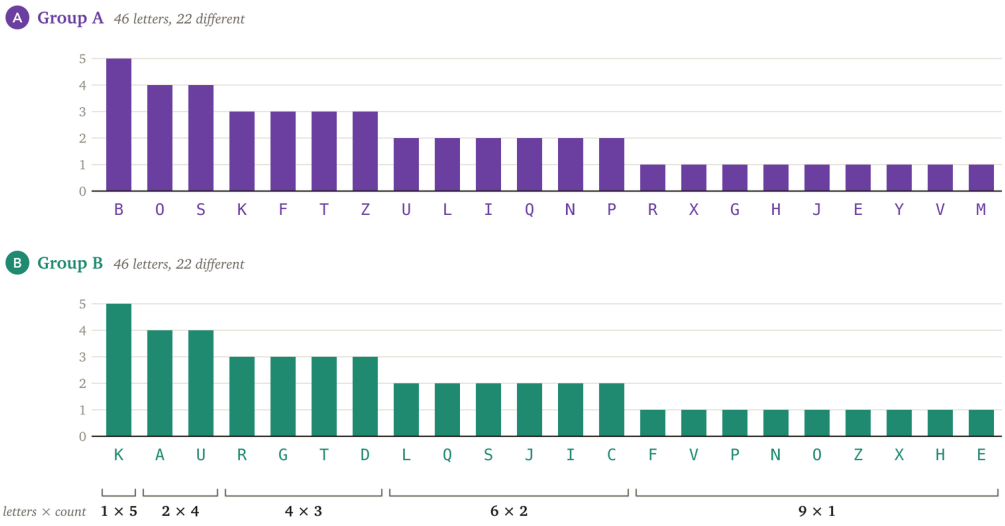
[1] The W split, alternating groups and matching frequency shapes were already described in an April 2016 answer by Bobson, crediting "void solid" and "dankuda" of the Kryptos Yahoo group. I did not know of it when I wrote this article: I reached the same observation independently.



- The groups appear alternately in the ciphertext: A|B|A|B|A|B.



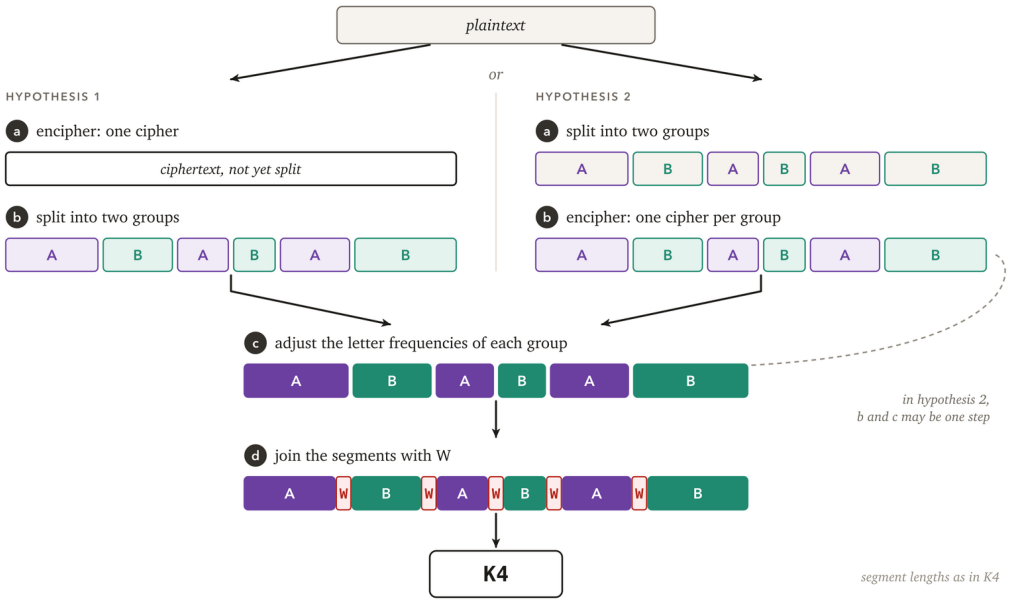
- The groups have identical letter frequency distribution shapes for different letters. In each group,
  - 1 letter appears 5 times,
  - 2 letters appear 4 times each,
  - 4 letters appear 3 times each,
  - 6 letters appear 2 times each, and
  - 9 letters appear 1 time each.



Based on these findings, I assume that:

- The passage has been enciphered by a mechanism that preserves the position of the words (substitution cipher). Indeed, a transposition of the whole text would not be compatible with such a segmentation: the latter requires preserving the original positions.
- The transformation of the plaintext into the ciphertext has been carried out using one of the following methods, both illustrating a **"segmentation and recombination" technique**:

    1. **(Hypothesis 1)** The plaintext has first been enciphered *(a)* and then split into segments divided into two groups *(b)*, adjusted to have the same letter frequency distribution shapes *(c)*, and recombined with intermediary letters 'W' *(d)*,
    2. **(Hypothesis 2)** The plaintext has first been split into segments divided into two groups *(a)*, then each segment has been enciphered using a technique specific to its group *(b)*; additionally, the groups have been adjusted to have the same letter frequency distribution shapes *(c)*, before being recombined with intermediary letters 'W' *(d)*. *(Note that, in that case, I do not exclude the possibility that the steps (b) and (c) could have been performed in one step).*



My Approach

While studying Enigma (hypothesizing it would help solve K4), I learned that separators represented punctuation marks as the machine only supported alphabetic characters. For instance, the doublet **"YY"** represented a dot. The same manual also lists "X = Full stop (end of sentence)", which is the hypothesis of the 2026 sequel. These separators were used for plaintext because of a technical constraint. *I pondered if this technique could have been replicated in ciphertext to misdirect analysts.* (Such separators are *delimiters*, an idea I later explored in a series of articles.)

To systematically explore this idea, I created **K4nundrum**.

This open-source tool splits K4 on different separators and analyzes the resulting segments. K4nundrum explores separators ranging from 'A' to 'Z' and uses them to split the ciphertext, subsequently permuting the resulting segments. Every permutation is distributed into different groups (from two groups for the whole ciphertext, up to one group per segment). Ultimately, K4nundrum examines each collection of groups to ascertain whether they exhibit identical letter distribution characteristics.

Using this process, **K4nundrum unveiled that splitting K4 based on the letter 'W' produces groups of segments with identical letter frequency distribution shapes** when permuted according to a specific arrangement (alternating groups). As mentioned above, **these two groups are { INFBNYPVTTMZFPK, OBKRUOXOGHULBSOLIFBB, TQSJQSSEKZZ } and { ATJKLUDIA, FLRVQQPRNGKSSOT, GDKZXTJCDIGKUHUAUEKCAR }**.

An Intentional Pattern?

This pattern is interesting: it is nonobvious, which would explain why it protected K4's plaintext so well. At the same time, it would have been relatively easy for the creator of Kryptos to implement (I attempt to demonstrate it with an illustrative example below). But couldn't it be just coincidental?

To understand how frequently such a pattern would occur in a string of text the length of K4, I have added a feature to K4nundrum that simulates "pseudo-K4s" (enabled with the `--sim` flag). These pseudo-K4s are random strings of 97 uppercase letters. (By definition, I cannot create a process that would generate perfect K4-like strings, as the underlying encryption method is unknown.) While these strings are not generated like the original K4, this simulation gives a general indication of how frequently this phenomenon can be noticeable *by an external observer*.

Simulations empirically indicate, based on the generation of more than one million pseudo-K4s, that this pattern naturally emerges less than 2% of the time. This probability is low but does not necessarily make this hypothesis a reality. Fortunately, several indicators demonstrate that K4's pattern has characteristics that make it less likely to occur just by chance.

First, the simulator tends to produce groups containing tiny segments (1 or 2 characters long). For instance, and typically:

```
> GKUCPFTQBJCKSEVMFWRLWKPEKCZHFFYNPLZZINWAFVKZEMJNRCAUYHUMITLKEZKQGZKBG
PIRRPUGKVSMHXASUIYPVJBZKFUKX

  Group 1:            BGPIRRPUG FU G PE QGZ UCPFTQBJC X ZEMJNRCAUYHUMITL
  Letter Freq.:       U:5 G:4 P:4 R:3 C:3 E:2 B:2 I:2 F:2 Z:2 M:2 J:2
                      T:2 Q:2 H:1 L:1 X:1 N:1 Y:1 A:1

  Group 2:            CZHFFYNPLZZINWAFV EZ SEVMFWRLW VSMHXASUIYPVJBZ
  Letter Freq.:       Z:5 V:4 F:4 W:3 S:3 H:2 E:2 P:2 L:2 M:2 N:2 Y:2
                      I:2 A:2 J:1 X:1 R:1 U:1 C:1 B:1
```

The random string, split at its separator K, produces segments such as 'EZ' or 'G' in this example.

If I assume that segments represent sequences of whole words, it is implausible that the creator of a ciphertext would use small segments. They would also needlessly

complicate the encryption process. When the simulator excludes tiny segments (fewer than three characters), ~0.6% of the pseudo-K4 passages can be split into appropriately sized groups with identical distribution shapes.

Second, K4's groups alternate. This additional constraint is even more rarely visible in the segments generated by the simulator: ~0.07% of the pseudo-K4s alternate. **When all these constraints are considered (identical letter frequency distribution shapes, appropriately sized segments, and alternating groups), only ~0.044% of the generated strings are similar to K4**. In this random-string model, that is about one generated string in 2,250. It does not estimate how many real ciphertexts Sanborn would have needed to create: the simulator does not model his unknown encryption method or design process.

Third, when I align the cribs with K4, the plaintext words are immediately adjacent to a 'W'. In other words, when mapped to the ciphertext, their first or last letter is directly next to a 'W': they start or end a segment. This observation could support the idea of segments representing sequences of whole words.



Suppose that segments are to be deciphered to (partial) sentences with complete words (the second segment, consisting of 15 characters, could, for instance, mean "… east-northeast **of** …"). In that case, the probability that such a pattern would emerge just by chance becomes even lower. Speculatively, it may be significant that Jim Sanborn, anxious to see his work finally deciphered, revealed words belonging to two *different* groups (BERLINCLOCK belonging to Group A, and EASTNORTHEAST to Group B)—as a kind of meta-clue.

These observations motivate my hypothesis that the pattern is intentional. The simulations do not establish that conclusion: their frequencies depend on a particular random-string model and on constraints chosen after observing K4. The hypothesis remains untested.

A Technique to Mask Letter Frequencies

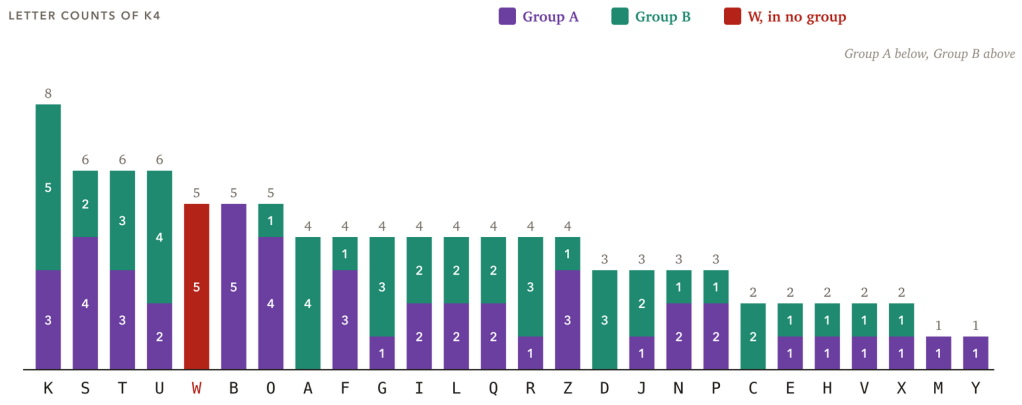
Fabricating Letter Frequencies

Why would Sanborn use such a technique?

First, as mentioned above, a segmenting and recombining technique conveniently produces nonobvious patterns. It is asymmetrical: it does not require a deep knowledge of cryptographic methods for its implementation; at the same time, it easily defeats classic cryptanalysis attacks by adding characters—the separators—that

are not mapped to the plaintext and misleading people to think that the ciphertext must be analyzed as a whole. In the context of K4, this is all the truer because the passage is so brief.

Second, this technique can mask the letter frequency by combining multiple letter frequencies. Let's take 'K' for example. It appears eight times in the original K4. By analyzing it under the lens of the segmentation technique, *these occurrences result from combining three occurrences in Group A and five in Group B*. By adjusting the letter frequency of one (or both) of the groups, it becomes possible to fabricate a new letter frequency distribution. Additionally, the letter separator pollutes the letter frequency distribution.



I do not know yet the exact mechanism employed to alter the distributions of K4's groups. It is, however, important *not* to assume that identical distribution shapes necessarily mean that the deciphered segments of Group A contain the same letters as the ones of Group B (which would be very unlikely for a natural language). On the one hand, this mechanism could have been applied to encrypted segments (second-order encoding). On the other hand, *intentional* spelling errors, similar to other passages like UNDERGR**U**UND (K2) or DESP**A**RA**TL**Y (K3), could have been introduced to slightly modify the letter frequencies. With this simple technique, one letter's occurrences are decremented while another's are incremented. For example, I**Q**LUSION (K1) decrements the Ls while incrementing the Qs. Similarly, superfluous letters could have been introduced in the style of **X**CANYOUSEEANYTHING**Q** (K3).

Illustration: Let's Encrypt Alice

I can test my hypothesis by applying this technique to a fictitious example.

Let's say I want to encrypt this passage from *Alice's Adventures in Wonderland*:

> "You promised to tell me your history, you know," said Alice.

The first step is to transform the sentence by removing the punctuation marks and the spaces and by capitalizing the letters:

"You promised to tell me your history, you know," said Alice.

↓ *capitals; no spaces, no punctuation*

YOUPROMISEDTOTELLMEYOURHISTORYYOUKNOWSAIDALICE

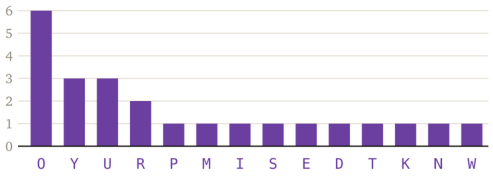
46 letters

Then, the sentence is divided into six segments, assigned alternately to two groups. Last, the letter frequencies of the respective groups are computed.

YOUPROMISEDTO (1) TELLME (2) YOUR (1) HISTORY (2) YOUKNOW (1) SAIDALICE (2)

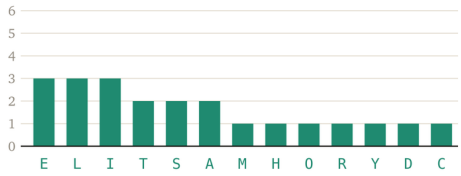
Group 1

YOUPROMISEDTO YOUR YOUKNOW



Group 2

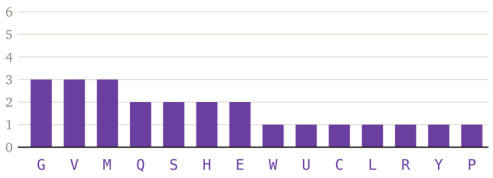
TELLME HISTORY SAIDALICE



The distributions are dissimilar. I want to reduce this difference. To that end, I encode the first group using standard Vigenère (obviously, I could have used a different cipher) and try several keys to improve the situation. The key `SECRET` quickly emerges as a good candidate ("YOUPROMISEDTO YOUR YOUKNOW" becomes "QSWGVHEMUVHMG CQLV RGYMESP"):

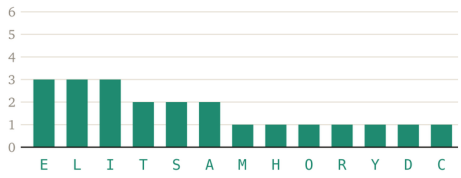
Group 1 Vigenère, key SECRET

QSWGVHEMUVHMG CQLV RGYMESP

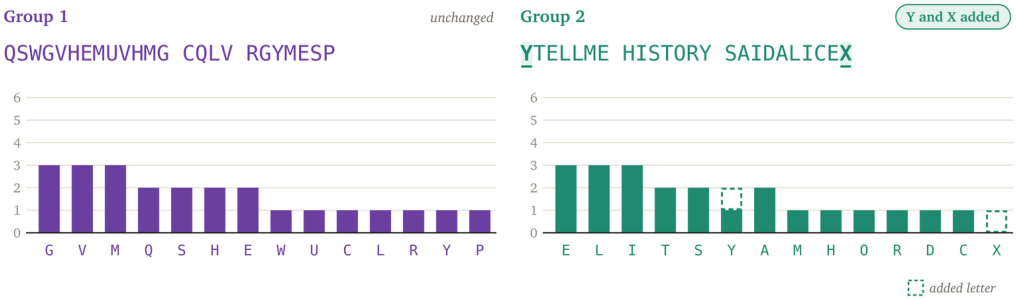


Group 2 *unchanged*

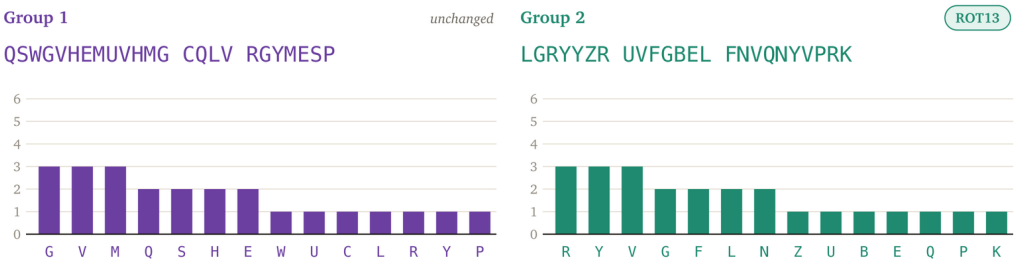
TELLME HISTORY SAIDALICE



The letter frequency distribution shapes are now similar but have yet to be identical. I need to adjust the *second* group to make them match. In this case, it is just a matter of adding letters (as the second group is shorter than the first). "TELLME HISTORY SAIDALICE" becomes "**Y**TELLME HISTORY SAIDALICE**X**". The original plaintext segments remain intelligible.



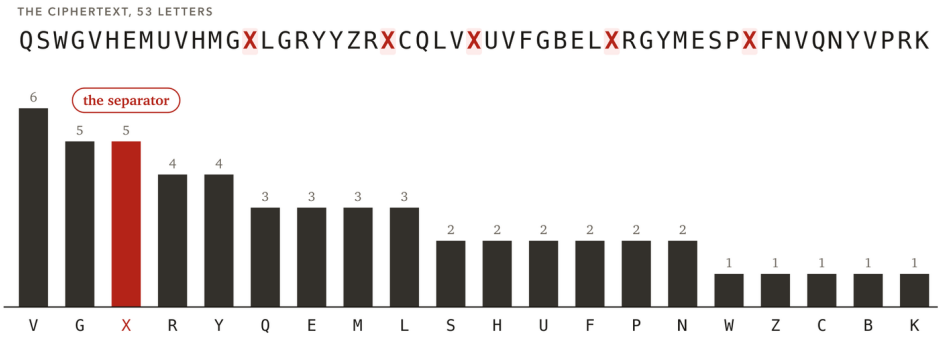
I obfuscate the second group for demonstration by merely applying a trivial ROT13 cipher, which preserves the distribution shapes ("**Y**TELLME HISTORY SAIDALICE**X**" becomes "LGRYYZR UVFGBEL FNVQNYVPRK").



The shapes of the distributions are identical. The last step is to identify a letter not present in the ciphertext segments ('X', for instance) and recombine the segments by alternating them:



The outcome is a ciphertext that cannot be attacked with classic methods because of the fabricated letter frequency.



Importantly, this example demonstrates that the two plaintext groups are not required to contain the same letters. Also, the separator that combines the segments is one of the most frequent letters in the ciphertext, even though it is the odd one out.

Conclusion

Once the technique is known, the separator is identified, and the correct groups are determined, deciphering this kind of code is a matter of understanding how each group has been "encoded" before attacking it with classic cryptanalysis methods.

This is the path I would like to follow to crack K4.

Series and Companion Articles

Kryptos K4 (glthr.com/k4)

#	Article	Address
1	**Kryptos K4: The 'W' as a Separator** (this article)	glthr.com/kryptos-k4-the-w-as-a-separator
2	Kryptos K4: The W-Rectangle and Its Half-Turn	glthr.com/kryptos-k4-a-rectangle-hidden-in-the-cipher

How to cite this article

> Guillaume Lethuillier, "Kryptos K4: The 'W' as a Separator", glthr.com, March 30, 2024.
> https://glthr.com/kryptos-k4-the-w-as-a-separator

For BibTeX:

```
@misc{lethuillier2024kryptos,
  author = {Lethuillier, Guillaume},
  title  = {Kryptos K4: The 'W' as a Separator},
  year   = {2024},
  month  = mar,
  url    = {https://glthr.com/kryptos-k4-the-w-as-a-separator},
  note   = {glthr.com}
}
```

Your questions and leads are welcome: kryptos@glthr.com

This document was typeset with LaTeX from the article published at https://glthr.com/kryptos-k4-the-w-as-a-separator, which remains the reference version (as updated on November 15, 2026).