Known Plaintext Attack (KPA):

1) Using KPA.java, I found the following:

Key: insane

Plaintext: Welcome to the Cryptography and Secure Development course 2024-2025, Miko. This is the secret message (i.e., plaintext) for you to be decrypted. Please make sure that this message is only designed for you by checking whether your ID number 2628565 is correct :) Good luck!

2) Different keys may produce the same two letters at the start (‘We’) due modular arithmetic being used, which means loop around 96 and so there is a chance of this looping around to ‘W’ and ‘e’ by chance. With this encryption method, two different keys can map to the same letters for the first few characters, although this will quickly diverge.

3) The modulus limits the character set to 96. And so, the chances of the first two characters being ‘We’ is (1/96)^2 in a uniform distribution. The chance of this happening is 0.0001085069444%. This can be rounded up to 0.0001%.

Ciphertext Only Attack (COA):

1) Using COA.java, I found the following:

Key: ganesh

Plaintext: 2025-02-19T15:58:15.0960958834 MailTo: 2628565O@student.gla.ac.uk. 8834 is a random number generated only for you, Osak, in the second task of the Project Assignment. In this Ciphertext Only Attack (COA) of the Project Assignment, you are expected to decode it with the assumption that this plaintext is English sentences; however, it can contain some special characters such as ! @ £ $ % ^ & * ( ). Finger-crossed!

2) I decided to use the two most common three-letter words in English: ‘the’ and ‘and’, with spaces either side of the word. ‘ the ‘ and ‘ and ‘ each consist of 5 characters. This means they have a 1.226433020E−10% chance of appearing in a random string. This is roughly 0.0000000001% or one in ten billion. And so, this means that this method would be unlikely to return a random string, and is likely to return valid English plaintext. Even in the unlilkely event of it returning a invalid plaintext, it would reduce the amount of strings enough to were a people could analyse it.

3) To do this theoretically, you have to find the unicity distane.The longest password in the ‘passwords’ list is ‘contortionist’, which is 14 characters long. All lowercase English letters as well as numbers are used = 36. From this, I assume the keyspace is 36^14. Assuming all of the characters see equal use, the entropy would be log2(36^14) = 72.37895002. The redundancy of English is 3.2. In order to calculate the unicity distance, we have to divide the entrophy of the keyspace by the plaintext redundancy. So 72.37895002 / 3.2 = 22.61842188. And so, 23 is the minimal length of ciphertext for there to be one decryption that leads to valid English plaintext.

In order to perform an experiment, I used a list of English words (https://github.com/dwyl/english-words/raw/refs/heads/master/words.txt), I decrypted the second given ciphertexts with all of the keys, and I checked how many decryptions start with words from the list of English words. Using this method, the longest valid English strings (excluding the correct decryptions) produced were 4 characters long.

The experiment was limited by the length of the key list given. A longer list would have provided more decryptions, making it more likely to see more English at the start of the decryptions at some point. Unicity distance works similarly to my experiment but gives a theoretical number as the number of keys approach infinity. The chance of getting the wrong decryption if the first 23 characters produce valid English should be 0%.