Challenge 1:

Upon completing an introductory course on cryptography, Tom, a Computer Science student, decided to utilize a classical cipher to safeguard certain secrets.
In an attempt to enhance the secrecy of his cipher text, he altered the resulting ciphertext, believing that it would be difficult for anyone who gains access to it to decipher and obtain the confidential information

The resulting secret is as follows:
KDEwMTAwMDExMTAxMDEwMTAwMTEwMDExMTAxMDAxMDAwMDExMTEwMDAwMTEwMTAwMTAxMTAxMTAwMTAxMDEwMCwxMDAxKQ

-> Given one of Tom's encrypted secrets, obtain the original plaintext secret that Tom tried to safeguard

* Note: All Tom's secrets are valid English words and sentences that only contain alphabetic characters [A-Za-z].

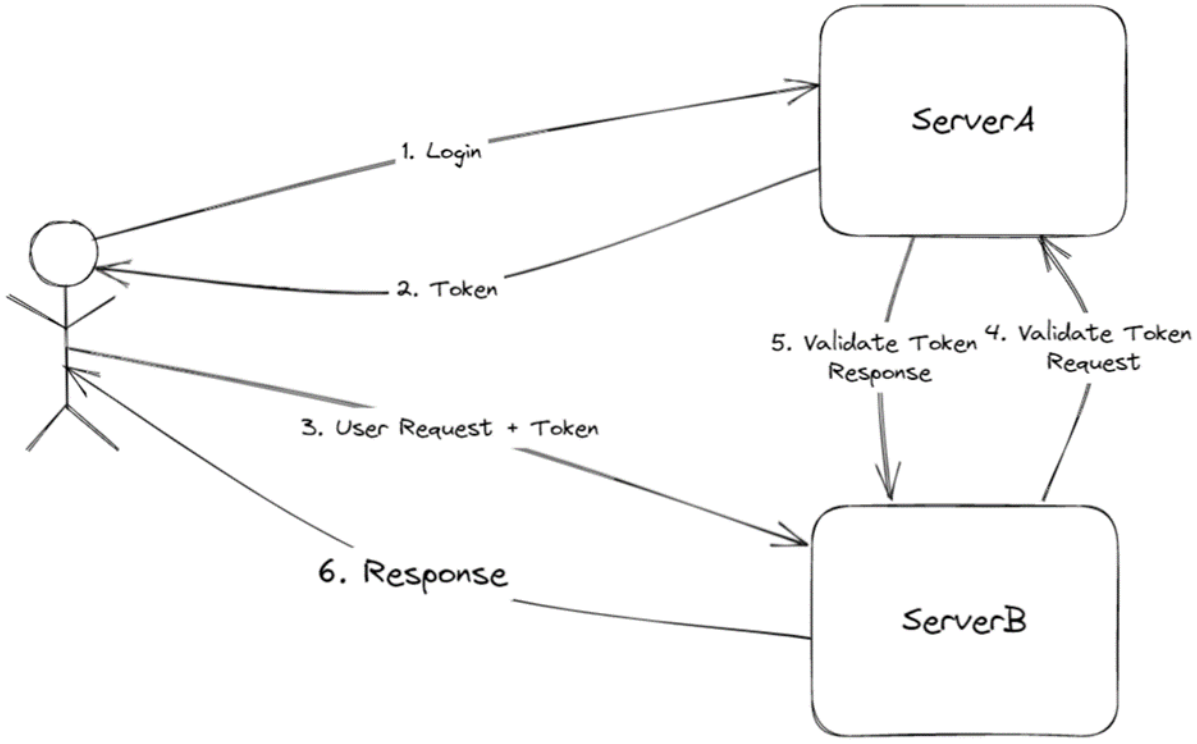
Example of supplied ciphertext:
KDEwMTAwMDExMTAxMDEwMTAwMTEwMDExMTAxMDAxMDAwMDExMTEwMDAwMTEwMTAwMTAxMTAxMTAwMTAxMDEwMCwxMDAxKQ

Expected solution:
The secret encrypted by Tom

##

Challenge 2:

In a backend environment consisting of serverA and serverB, serverA is responsible for authenticating users by providing a session token (JWT) upon validating their credentials. However, when the user interacts with serverB, it cannot verify the token's legitimacy due to it being signed using a symmetric encryption algorithm, and only serverA has the key required for signing and validation.



To mitigate this issue, the maintainers of serverA proposed using Asymmetric Encryption (using RS256 algorithm) to sign and validate tokens. This approach involves token signing with a secret key kept on serverA, and token validation with a public key embedded as JWK within the JWT header. Consequently, any server can authenticate the token's legitimacy without requiring a call to serverA, reducing latency.

-> Given a JWT, your task is to modify the token by setting the "admin" attribute to "true" so that serverB will be deceived into believing that you have admin privileges.

Example of supplied JWT:

eyJraWQiOiI5MmM1YTU1MS0yNjFlLTQ5OTktOGQyMy0wMjM1YTg2NTM1NjEiLCJ0eXAiOiJK
V1QiLCJhbGciOiJSUzI1NiIsImp3ayI6eyJrdHkiOiJSU0EiLCJlIjoiQVFBQiIsImtpZCI6IjkyYzVhNT
UxLTI2MWUtNDk5OS04ZDIzLTAyMzVhODY1MzU2MSIsIm4iOiJ3SUZuVmp1WUpEQzAwZEhi
MTZXOUhTaHZrelZOcS1wQkV5Q3Q1djJYQXBsa08xVFQxdlV2ZDBsdnc3RldqU2xkeU5iLW11
Wk1jUjZDSjYxd2U4dnVTa3p0S0JmelUzN1FZbWhwQ1YzUFBxRFdmdTNMSlJFZEFHckFsOGt
YMkxpTjU0YWN2RENFNVRlYWpwR1RscEJ6ZFUyNHhMaHpua1NmWUhjWnhlRjVrSzFHS3g2
SUFqRm5fUDFCU1BvMGg5cGo4Rk83Z2pndExnODM1NGs1YXREV0dRUjF1anctUzZSLUFaU
nJIRldvUEZBclphUmd4bHo4VktBREdxRm12X1JDc19FM2FYZnJoTjhMT0Z3cU9pYlk2NklERT
Raa3hrVEYtQ3ZNWHVSMVRuaDR6aG5uY2VLRGhYZy1ITjhCbGN1cmhNbjA5cFJGelV2eVpM
VFhOUExvdFEifX0.eyJpc3MiOiJodHRwczovL3NlcnZpY2VBLmVudjo4MDgwIiwiYXVkIjoiYWNjb
3VudCIsImlkIjoiNTAxYTFiZjQtYjUyMy00OTNjLWI0NTctYWVmOTM1YTI0MWJhIiwic2NvcGUiOi
AicHJvZmlsZSIsICJuYW1lIjoiQm9iIE1hcmxleSIsICJlbWFpbCIgOiJib2IubWFybGV5QGNvcnAu
Y29tIiwiYWRtaW4iOiJmYWxzZSJ9.S4ttPFFFBi1HjV_DB1CdTAyIgPgFVlk0ssGyjWDP-8WlUvB
BUSK_3ej_DA2TQRKXxlrkdua-fw5FNQ1mYH4yJm5fGL-6O65ZeksHla1NZWdiVRYkcoCCNzvI
16KEZnzKOO9-tLsjMrblPZWCl-WLy5B1bfqSxDvpGPCZV0vwnotCTjmsQdPxBnlC1_3-3mOi6rE
DSxTnb_ew5B8nF8_dQuTjsdV44m85mObFxkowhnwq8QS7ENbsJAvt6KaneRMVYiHwKCJwJe
dAVR2wh5V5hB7k8emQACdWIG0hImf-FWSmaTsdVwOpaPC5Q0VrYNIbWq9q1HgNfvq4J2q3
PhsUgw

Expected solution:

A valid JWT with "admin" attribute set to “true”

##

Challenge 3:

A network security analyst received an alert for a suspicious activity where a certain machine was sending multiple DNS queries to an unknown DNS server (188.68.45.12).
Upon examining sample packets, he suspected a data exfiltration process and immediately disabled network access for the machine to initiate an investigation.
After analyzing the packets, the analyst was certain that a malicious actor was leaking the company's data. However, he could not determine the exact nature of the leaked information, as it was being sent in an obfuscated manner that he could not decipher.

-> Your task is to examine a PCAP file and uncover the complete secret that was leaked by the malicious actor.

Example of supplied PCAP:
PCAP file encoded in Base64 format.

Expected solution:
The secret leaked by the malicious actor

Challenge 4:

You have been presented with a challenging security task - developing a solution to bypass the CAPTCHA generated by Amazon.com such as the one accessible via the URL "https://www.amazon.com/errors/validateCaptcha"
The objective of this challenge is to develop a solution that can evade the security measures put in place by Amazon.com and successfully solve the CAPTCHA, without requiring any human intervention.

*Note: Solution can make use of open source libraries.

Example of supplied CAPTCHA:
Numpy array representing the CAPTCHA image.

Expected solution:
The solution for the CAPTCHA