A Design Issue:

Note: The design issue give below was part of Midterm II in Fall 2019. A solution will be provided by Thursday, November 19, 2020. Try your luck before that!

Three junior architects were discussing a design requirement for their recent software product. The design requirement goes like this: In order to send and receive a large sensitive information over the internet to a remote server, a compression and encryption modules are to be employed for this purpose. This means the data needs to be encrypted and then compressed before sending it over and upon receiving, the data needs to be uncompressed and decrypted for further processing. The discussion was extended to a debate and the architects were told to present their design solutions.

Hamza was first to present his design. He strongly believed that Adapter pattern is to be employed because then the developers have the liberty to use any algorithm or replace it in case an improved version is available in the market.

Hamza is correct because the problem discussed in the first paragraph focused on these two modules encryption and compression. In most of the APIs there are mainly single class which perform such algorithms, rest of the classes in the APIs are for setting up parameters for such large calculations. Therefore, Adapter pattern is the right way to go.

On the other hand, Haniya explained that not only these 2 classes are needed in the software, but there are couple of more classes from different APIs they are using, such as event loggers to log each transaction and XML convertor to transfer the data into XML format for another subsystem. Therefore, she thinks that Façade pattern would be much suitable here because then they do not have to create separate adapters for each of the API classes but rather use just one class which caters for several subsystems.

Haniya's solution could also be correct only if we think that additional items are needed. There is also a case where she found that additional items are nice to have. However, if we focus on the problem above, Façade could not be the best option.

Hammad has an opinion to oppose both of the above options. He presented that Bridge pattern is clearly applied here because Encryption and Compression are specifically for each record, whereas all other external modules are applied in general on the data records. He has an opinion that Encryption and Compression are aspects and not the core responsibilities of data records.

Hammad is mostly incorrect in identifying the problem because Bridge pattern is generally not considered to be applied with external API. This means, the pattern would only be used if we are developing encryption and compression for its several different implementations, and design is prepared in such a way to bifurcate abstraction with its implementation. At the very least the problem above does not consider any separation of abstraction and implementation. Therefore, this design will not pass.

What is your opinion given the summary provided above? Whose design is better? Do you have even better design idea to be employed here other than the above ones? Provide strong reasons for your option. Do you think that the information is enough to conclude a design decision? In the case you think the given information is incomplete, explain with strong reasons why do you think we cannot decide about the design to be employed?