NOTE: SCENARIO IS WHAT PUZZLE TAKER SEES.

SCENARIO:

You are developing a secure texting system. The requirement asks to encrypt text messages before transferring them over network. The `encrypt` method takes three arguments: `String alg` (an encryption algorithm), `Key key` (a cryptographic key used for encryption), `String text` (a string value to be encrypted), and returns a `byte` array representing the ciphertext of the original (plain) text. Consider the snippet of code below and answer the following questions, assuming that the code has all required permissions to execute.

```
01   // OMITTED: Import whatever is needed.
02   public final class CryptoUtils {
03     public static byte[] encrypt (String alg, Key key, String text)
04         throws GeneralSecurityException {
05       // Create a cipher
06       Cipher cipher = Cipher.getInstance(alg);
07       cipher.init(Cipher.ENCRYPT_MODE, key);
08
09       // Encrypt the data
10       byte[] input = text.getBytes();
11       byte[] output = new byte[input.length];
12
13       int length = cipher.update(input, 0, input.length, output, 0);
14       length += cipher.doFinal(output, length);
15       return output;
16     }
17   }
```

Questions:

1. What will the `encrypt` method do when executed?

2. If the `encrypt` method gets called with a valid algorithm and valid key and a non empty string, which one statement is correct?

a. Depending on the key, the `encrypt` method may encrypt the text correctly.
b. Depending on the algorithm, the `encrypt` method may encrypt the text correctly.
c. The `encrypt` method never encrypts the text correctly.
d. The `encrypt` method always encrypts the text correctly.
e. The encrypt method will lead to a crash in the program where it is invoked.

[Other statistical questions will be imported here while creating survey.]

NOTE: ANSWER IS TO BE SHOWN TO PUZZLE TAKER AT THE END OF SESSION.

ANSWER:
b
The code snippet, as is, does not encrypt the data properly. Depending on the algorithm, the output of the encryption process (the `output` array) might be longer than the input of the encryption process (the `input` array). According to the API documentation the `getOutputSize` method of a `Cipher` object returns the length that an output buffer needs to have in order to hold the result of the next `update` or `doFinal` operation. Thus to make sure that the `output` array has enough space, it must be instantiated by the returned value of `cipher.getOutputSize(bytes.length)`.

NOTE: THE REST OF DOCUMENT CONTAINS EXTRA INFORMATION FOR THE PROJECT RESEARCHERS. IT IS NOT TO BE SHOWN TO PUZZLE TAKERS.

TAGS:
java, cryptography, cipher, invalid-object-initialization, api-protocol-usage

CATEGORIES:
Blindspot - YES
Type - Crypto
Number of distinct functions - 6
Number of total functions - 6
Blindspot function - `Cipher.getOutputSize()`
Function call omitted - YES
Blindspot type - Omitted function call
Number of Parameters in the blindspot function - 1 parameters
Cyclomatic complexity - 2

Blindspot, Crypto, 5 functions, 2 parameters

NAME:
Cipher class, getOutputSize method - Provides the functionality of a cryptographic cipher for encryption and decryption. It forms the core of the Java Cryptographic Extension (JCE) framework.

DESCRIPTION:
The creation and use of a `Cipher` object follows a simple pattern. You create one using `Cipher.getInstance()`, initialize it with the mode you want using `cipher.init()`, feed the input data in while collecting output at the same time using `cipher.update()`, and then finish off the process with `cipher.doFinal()`. Depending on the algorithm, the output length is always equal to or greater than the input's. It always better to ask the cipher object how big the length of output will be.

BLINDSPOT:

The fact that the length of encryption output (cipher text) is sometimes longer than the original input (plain text) may cause a blindspot. According to the API documentation, to get the correct length of output, programmer has to call the `getOutputSize` method of the cipher object. It is a common mistake to assume that the length of output is the same as input's.

CORRECT USE EXAMPLE:

```
import javax.crypto.*;
public class Application {
    public static void main ( String... args ) throws Exception {
        byte[] data = …; // Get some data to encrypt

        SecretKey key = …; // Create a key!
        Cipher c = Cipher.getInstance(…); // Create a cipher
        c.init(Cipher.ENCRYPT_MODE, key);

        System.out.println("Start encryption...!");

        // byte[] encrypted = new byte[data.length]; // Invalid!
        byte[] encrypted = new byte[c.getOutputSize(data.length)];

        int length = c.update(date, 0, data.length, encrypted, 0);
        length += c.doFinal(encrypted, length);

        System.out.println("Done!");
    }
}
```

MORE INFORMATION:
To see the correct way of use of the API look at the JX23-Cipher.getOutputSize puzzle.

REFERENCES:
1. Cipher