Gabriel Corella
Applied Cryptography
February 23, 2020

Question 4: Can you prove that the construction you implemented in question 3 above is semantically secure? In your answer, also discuss any assumptions you have made.

When using Java, the class java.security.* is used as a cryptographically strong random number generator (RNG). Additionally, SecureRandom must produce non-deterministic output.

```
public interface SecureRandomParameters()
```

- A marker interface for parameters used in various SecureRandom methods.

This implementation supports the Hash_DRBG and HMAC_DRBG mechanisms with DRBG algorithm SHA-224, SHA-512/224, SHA-256, SHA-512/256, SHA-384 and SHA-512, and CTR_DRBG (both using derivation function and not using derivation function) with DRBG algorithm AES-128, AES-192 and AES-256.

When using these methods or function calls, callers may also invoke the generateSeed method to generate a given number of seed bytes (to seed other random number generators, for example):

```
byte seed[] = random.generateSeed(20);
```

To ensure protection, callers can instead generate a new DRGB instead of calling the reseed() function.

$$\mathbf{E(K, R, P) = (DRBG(KR) \oplus P, R}$$

Where here, R is a string randomly chosen for each new encryption and given to a DRBG along with the key (K || R denotes the string consisting of Kfollowed by R). I believe the system built is semantically secure.