Correctness vs security properties

Most of the definitions we'll see in this class are either *correctness* properties (what SHOULD happen when nobody's trying to break anything) or *security* properties (what someone attacking the system SHOULDN'T be able to do).

Example correctness property If you know the combination, it should be easy to open the safe.

Example of a security property Assuming the combination was chosen randomly, if you don't know the combination you shouldn't be able to open the safe faster than by trying every combination.

When making security properties formal, we need to specify

- initialization (lock the safe with a randomly chosen combination)
- the win condition for the adversary (opening the safe faster than by trying every combination)
- any special tools or abilities we assume the adversary has (none in this example)

Activity 1: Correctness vs security properties

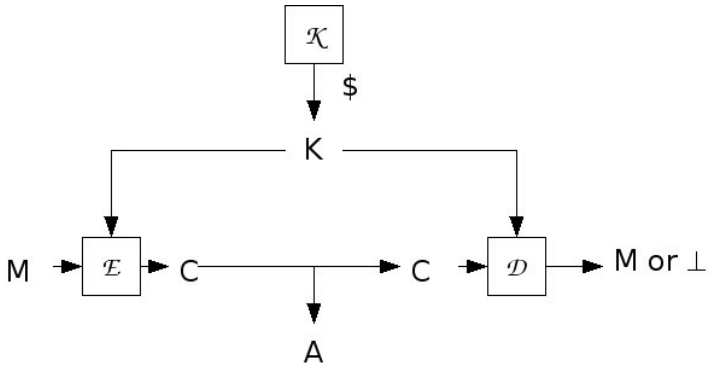
In groups, decide whether each of the following is a correctness property or a security property. For each security property, identify the initialization, the win condition, and any special abilities the adversary has. (The below examples are informal, so these might not all be mentioned explicitly as they would be in a formal definition.)

1. For an encryption scheme, if you encrypt a message using key K, and then you decrypt the resulting ciphertext using key K, you should always get back the original message.
2. If the random number generator is properly seeded, it should be impossible (or at least impractical) to predict the next output, even if you've seen all the previous outputs.
3. The safe should be able to withstand someone trying to cut it open with a hacksaw for at least two hours.
4. Once a user has signed up for 2FA, they can log in using their password and their six-digit one-time password code.
5. Even if the doctor's laptop is stolen, patient medical records must remain secret.
6. Even if one or more protest organizers have their phones seized, authorities shouldn't be able to learn the identities of the other organizers.

Activity 2: Make your own security definitions

In future lectures we'll see many security definitions that cryptographers have come up with and found to be useful. In this activity you'll make up some definitions of your own!

Recall the following picture from class:



(Adversary)

1. Do you remember the *correct decryption requirement* for symmetric encryption? Is that a security property or a correctness property?

2. Try coming up with some security definitions of your own! What are some possible "win conditions"? What are some special abilities that the adversary could have? What initialization are you assuming?

3. Can you think of any use-cases for encryption where someone would need a security property that *isn't* captured by the definition you came up with?