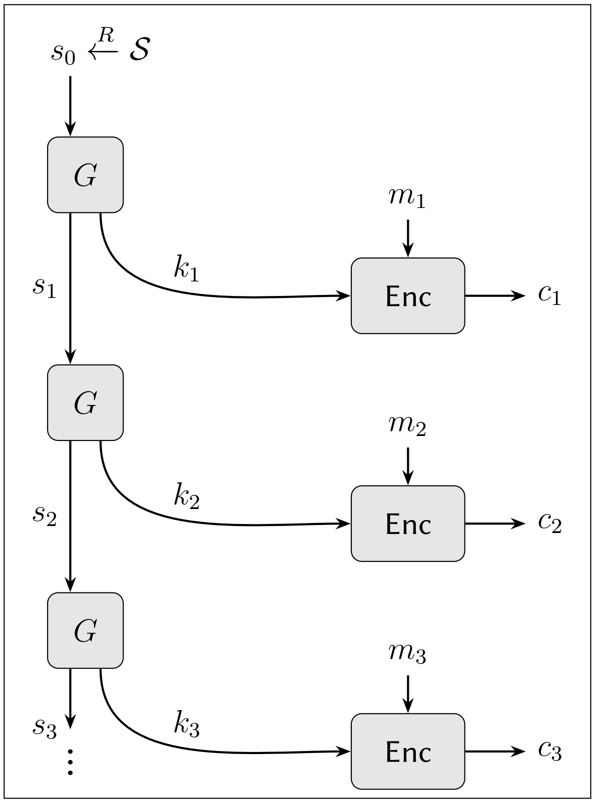
Solutions to Discussion 5

Problem 1.

The Signal protocol is the most widely used end-to-end encrypted messaging protocol in the world, used by over a billion people daily. It's the core cryptography underneath WhatsApp, Google Messages, Facebook Messenger, and the eponymous Signal app.

An important property Signal provides is *forward secrecy*: each time a message is sent, the encryption key is changed so that message remains hidden from this moment forward, even if an attacker obtains future encryption keys. The mechanism Signal uses to update the keys is called a *symmetric ratchet*[1]. The core ingredients to a symmetric ratchet are a PRG G and a semantically secure cipher $(\mathsf{Enc}, \mathsf{Dec})$.



Symmetric Ratchet

```
$s_0 \overset{R}{\leftarrow} \mathcal{S}$
for $i \twoheadleftarrow 1$ to $\infty$:
  $(s_i, k_i) \twoheadleftarrow G(s_{i-1})$
  Delete $s_{i-1}$ from memory
  Use $k_i$ to encrypt/decrypt the $i$-th message
  Delete $k_i$ from memory
```

Figure 1: The Signal symmetric ratchet

Explanation of Figure 1 . When Alice and Bob want to communicate with a symmetric ratchet, they begin by agreeing on a uniformly random PRG seed s_0. If Alice is sending the first message, she runs the seed s_0 through G to obtain a new seed s_1 as well as a key k_1. She encrypts her message under k_1 with Enc and send the ciphertext to Bob. Now Bob runs his copy of s_0 through G to learn the same key k_1 and decrypt the message. This process can repeat essentially forever: with each new message, Alice and Bob use the PRG to update their state and produce a fresh encryption key. Every time Alice or Bob

[1] The full Signal protocol is sometimes called the double ratchet protocol, because it interleaves a symmetric ratchet with another mechanism called an asymmetric ratchet. The asymmetric ratchet provides backwards security, meaning that messages are secure from key compromises that occurred in the past.

updates their state with G, they delete the previous state and key from memory. This ensures that if they are compromised, the attacker only learns the current state.

This mechanism is called a *ratchet* because it's easy to advance the state forward, but it's hard to recover a previous state once you've deleted it from memory.

Problem statement. In this problem you'll show that the symmetric ratchet has a property necessary for forward secrecy: specifically, that there is no efficient way to learn a past key given a later state and key. (This property is necessary, but not sufficient, because what you really want is that past encryptions are still semantically secure; but we are giving you a simpler problem here.) Let $G : \{0,1\}^{\ell} \to \{0,1\}^{2\ell}$ be a secure PRG. For all $n \in \mathbb{N}$, let $G^n : \{0,1\}^{\ell} \to \{0,1\}^{(n+1)\ell}$ be the n-wise sequential composition of G, i.e.

```
$G^n(s)$
$s_0 \twoheadleftarrow s$
for $i \twoheadleftarrow 1$ to $n$:
    $(k_i, s_i) \twoheadleftarrow G(s_{i-1})$
output $(k_1, \ldots, k_n, s_n)$
```

Prove that there does not exist a PPT algorithm $\mathcal{A}$ such that $\Pr[\mathcal{A}(k_n, s_n) == k_{n-1}]$ is non-negligible, when k_n, s_n, and k_{n-1} are values produced by G^n with a random seed.

You can use the fact that G^n is a secure PRG, as proven in section 3.4.2 of the textbook. Suppose there exists such an algorithm $\mathcal{A}$, and use it to construct an adversary that breaks the PRG game for G^n.

Solution. Suppose $\mathcal{A}$ is a PPT algorithm where $\Pr[\mathcal{A}(k_n, s_n) == k_{n-1}]$ is non-negligible for k_n, s_n, k_{n-1} produced by G^n with a random seed. Let $p = \Pr[\mathcal{A}(k_n, s_n) == k_{n-1}]$. We'll construct an adversary $\mathcal{B}$ that breaks the PRG security of G^n as follows:

- $\mathcal{B}$ receives a string r from the challenger and parses it as $(k_1, \ldots, k_n, s_n) \twoheadleftarrow r$.
- $\mathcal{B}$ computes $k' = \mathcal{A}(k_n, s_n)$, and outputs 1 iff $k' == k_{n-1}$.

In experiment 0, the string r is equal to $G^n(s)$ for some randomly chosen s. Therefore, the probability that $\mathcal{A}$ outputs k_{n-1} is p, which is non-negligible by assumption.

In experiment 1, r is sampled uniformly from $\{0,1\}^{(n+1)\ell}$. In this case, the distribution of k_{n-1} is uniform over $\{0,1\}^{\ell}$, and independent of s_n and k_n. The probability that k_{n-1} equals the output of $\mathcal{A}$ is $\frac{1}{2^{\ell}}$, which is negligible. We have that

$$\begin{aligned}\text{PRGadv}[\mathcal{B}, G^n] =& \big| \Pr[W_0 \text{ in PRG game for } G^n] - \Pr[W_1 \text{ in PRG game for } G^n] \big| \\ =& \big| p - \frac{1}{2^{\ell}} \big|,\end{aligned}$$

which is non-negligible in ℓ, since a non-negligible function minus a negligible function is non-negligible. Therefore $\mathcal{B}$ breaks the PRG security of G^n.